POK(kernelpart)

Generated by Doxygen 1.8.3.1

Fri Aug 2 2013 11:46:09

Contents

1 Data Structure Index **1**

1.1 Data Structures 1

2 File Index **3**

2.1 File List 3

3 Data Structure Documentation **7**

3.1 __attribute__ Struct Reference 7

3.1.1 Detailed Description 8

3.1.2 Field Documentation 8

3.1.2.1 available 8

3.1.2.2 back_link 8

3.1.2.3 base 8

3.1.2.4 base_high 8

3.1.2.5 base_low 8

3.1.2.6 cr3 8

3.1.2.7 cs 8

3.1.2.8 d 9

3.1.2.9 dpl 9

3.1.2.10 ds 9

3.1.2.11 eax 9

3.1.2.12 ebp 9

3.1.2.13 ebx 9

3.1.2.14 ecx 9

3.1.2.15 edi 9

3.1.2.16 edx 9

3.1.2.17 eflags 9

3.1.2.18 eip 9

3.1.2.19 es 9

3.1.2.20 esi 10

3.1.2.21 esp 10

3.1.2.22 esp0 10

3.1.2.23 esp1 . . . 10
3.1.2.24 esp2 . . . 10
3.1.2.25 fs . . . 10
3.1.2.26 granularity . . . 10
3.1.2.27 gs . . . 10
3.1.2.28 io_bit_map_offset . . . 10
3.1.2.29 ldt . . . 10
3.1.2.30 limit . . . 10
3.1.2.31 limit_high . . . 10
3.1.2.32 limit_low . . . 11
3.1.2.33 offset_high . . . 11
3.1.2.34 offset_low . . . 11
3.1.2.35 op_size . . . 11
3.1.2.36 padding . . . 11
3.1.2.37 present . . . 11
3.1.2.38 res0 . . . 11
3.1.2.39 res1 . . . 11
3.1.2.40 s . . . 11
3.1.2.41 segsel . . . 11
3.1.2.42 ss . . . 11
3.1.2.43 ss0 . . . 11
3.1.2.44 ss1 . . . 12
3.1.2.45 ss2 . . . 12
3.1.2.46 trace_trap . . . 12
3.1.2.47 type . . . 12
3.2 context_t Struct Reference . . . 12
3.2.1 Detailed Description . . . 13
3.2.2 Field Documentation . . . 13
3.2.2.1 __esp . . . 13
3.2.2.2 back_chain . . . 13
3.2.2.3 cr . . . 13
3.2.2.4 cs . . . 13
3.2.2.5 eax . . . 13
3.2.2.6 ebp . . . 13
3.2.2.7 ebx . . . 13
3.2.2.8 ecx . . . 14
3.2.2.9 edi . . . 14
3.2.2.10 edx . . . 14
3.2.2.11 eflags . . . 14
3.2.2.12 eip . . . 14

3.2.2.13 esi . . . 14
3.2.2.14 lr . . . 14
3.2.2.15 pad . . . 14
3.2.2.16 r13 . . . 14
3.2.2.17 r14 . . . 14
3.2.2.18 r15 . . . 14
3.2.2.19 r16 . . . 14
3.2.2.20 r17 . . . 15
3.2.2.21 r18 . . . 15
3.2.2.22 r19 . . . 15
3.2.2.23 r2 . . . 15
3.2.2.24 r20 . . . 15
3.2.2.25 r21 . . . 15
3.2.2.26 r22 . . . 15
3.2.2.27 r23 . . . 15
3.2.2.28 r24 . . . 15
3.2.2.29 r25 . . . 15
3.2.2.30 r26 . . . 15
3.2.2.31 r27 . . . 15
3.2.2.32 r28 . . . 16
3.2.2.33 r29 . . . 16
3.2.2.34 r30 . . . 16
3.2.2.35 r31 . . . 16
3.2.2.36 sp . . . 16
3.2.2.37 unused_lr . . . 16
3.3 cpio_bin_header Struct Reference . . . 16
3.3.1 Detailed Description . . . 16
3.3.2 Field Documentation . . . 17
3.3.2.1 c_dev . . . 17
3.3.2.2 c_filesize . . . 17
3.3.2.3 c_gid . . . 17
3.3.2.4 c_ino . . . 17
3.3.2.5 c_magic . . . 17
3.3.2.6 c_mode . . . 17
3.3.2.7 c_mtime . . . 17
3.3.2.8 c_namesize . . . 17
3.3.2.9 c_nlink . . . 17
3.3.2.10 c_rdev . . . 17
3.3.2.11 c_uid . . . 17
3.4 cpio_file Struct Reference . . . 18

3.4.1 Detailed Description . . . 18
3.4.2 Field Documentation . . . 18
3.4.2.1 cpio_addr . . . 18
3.4.2.2 cpio_fmt . . . 18
3.4.2.3 curr_fileaddr . . . 18
3.4.2.4 curr_filename . . . 18
3.4.2.5 curr_filename_len . . . 18
3.4.2.6 curr_filesz . . . 18
3.4.2.7 curr_header . . . 18
3.4.2.8 next_header . . . 19
3.5 Elf32_Ehdr Struct Reference . . . 19
3.5.1 Detailed Description . . . 19
3.5.2 Field Documentation . . . 19
3.5.2.1 e_ehsize . . . 19
3.5.2.2 e_entry . . . 19
3.5.2.3 e_flags . . . 19
3.5.2.4 e_ident . . . 19
3.5.2.5 e_machine . . . 20
3.5.2.6 e_phentsize . . . 20
3.5.2.7 e_phnum . . . 20
3.5.2.8 e_phoff . . . 20
3.5.2.9 e_shentsize . . . 20
3.5.2.10 e_shnum . . . 20
3.5.2.11 e_shoff . . . 20
3.5.2.12 e_shstrndx . . . 20
3.5.2.13 e_type . . . 20
3.5.2.14 e_version . . . 20
3.6 Elf32_Phdr Struct Reference . . . 20
3.6.1 Detailed Description . . . 21
3.6.2 Field Documentation . . . 21
3.6.2.1 p_align . . . 21
3.6.2.2 p_filesz . . . 21
3.6.2.3 p_flags . . . 21
3.6.2.4 p_memsz . . . 21
3.6.2.5 p_offset . . . 21
3.6.2.6 p_paddr . . . 21
3.6.2.7 p_type . . . 21
3.6.2.8 p_vaddr . . . 21
3.7 interrupt_frame Struct Reference . . . 22
3.7.1 Detailed Description . . . 22

3.7.2 Field Documentation . . . 22
3.7.2.1 __esp . . . 22
3.7.2.2 cs . . . 22
3.7.2.3 ds . . . 22
3.7.2.4 eax . . . 22
3.7.2.5 ebp . . . 22
3.7.2.6 ebx . . . 23
3.7.2.7 ecx . . . 23
3.7.2.8 edi . . . 23
3.7.2.9 edx . . . 23
3.7.2.10 eflags . . . 23
3.7.2.11 eip . . . 23
3.7.2.12 error . . . 23
3.7.2.13 es . . . 23
3.7.2.14 esi . . . 23
3.7.2.15 esp . . . 23
3.7.2.16 ss . . . 23
3.8 pok_aout_symbol_table_t Struct Reference . . . 24
3.8.1 Detailed Description . . . 24
3.8.2 Field Documentation . . . 24
3.8.2.1 addr . . . 24
3.8.2.2 reserved . . . 24
3.8.2.3 strsize . . . 24
3.8.2.4 tabsize . . . 24
3.9 pok_elf_section_header_table_t Struct Reference . . . 24
3.9.1 Detailed Description . . . 25
3.9.2 Field Documentation . . . 25
3.9.2.1 addr . . . 25
3.9.2.2 num . . . 25
3.9.2.3 shndx . . . 25
3.9.2.4 size . . . 25
3.10 pok_lockobj_attr_t Struct Reference . . . 25
3.10.1 Detailed Description . . . 25
3.10.2 Field Documentation . . . 25
3.10.2.1 initial_value . . . 25
3.10.2.2 kind . . . 26
3.10.2.3 locking_policy . . . 26
3.10.2.4 max_value . . . 26
3.10.2.5 queueing_policy . . . 26
3.11 pok_lockobj_lockattr_t Struct Reference . . . 26

3.11.1 Detailed Description . . . 26
3.11.2 Field Documentation . . . 26
3.11.2.1 lock_kind . . . 26
3.11.2.2 obj_kind . . . 26
3.11.2.3 operation . . . 26
3.11.2.4 time . . . 27
3.12 pok_lockobj_t Struct Reference . . . 27
3.12.1 Detailed Description . . . 27
3.12.2 Field Documentation . . . 27
3.12.2.1 current_value . . . 27
3.12.2.2 eventspin . . . 27
3.12.2.3 initialized . . . 27
3.12.2.4 is_locked . . . 27
3.12.2.5 kind . . . 27
3.12.2.6 locking_policy . . . 28
3.12.2.7 max_value . . . 28
3.12.2.8 queueing_policy . . . 28
3.12.2.9 spin . . . 28
3.12.2.10 thread_state . . . 28
3.13 pok_memory_map_t Struct Reference . . . 28
3.13.1 Detailed Description . . . 28
3.13.2 Field Documentation . . . 28
3.13.2.1 base_addr_high . . . 28
3.13.2.2 base_addr_low . . . 28
3.13.2.3 length_high . . . 29
3.13.2.4 length_low . . . 29
3.13.2.5 size . . . 29
3.13.2.6 type . . . 29
3.14 pok_module_t Struct Reference . . . 29
3.14.1 Detailed Description . . . 29
3.14.2 Field Documentation . . . 29
3.14.2.1 mod_end . . . 29
3.14.2.2 mod_start . . . 29
3.14.2.3 reserved . . . 29
3.14.2.4 string . . . 30
3.15 pok_multiboot_header_t Struct Reference . . . 30
3.15.1 Detailed Description . . . 30
3.15.2 Field Documentation . . . 30
3.15.2.1 bss_end_addr . . . 30
3.15.2.2 checksum . . . 30

3.15.2.3 entry_addr . . . 30
3.15.2.4 flags . . . 30
3.15.2.5 header_addr . . . 30
3.15.2.6 load_addr . . . 31
3.15.2.7 load_end_addr . . . 31
3.15.2.8 magic . . . 31
3.16 pok_multiboot_info_t Struct Reference . . . 31
3.16.1 Detailed Description . . . 31
3.16.2 Field Documentation . . . 31
3.16.2.1 aout_sym . . . 31
3.16.2.2 boot_device . . . 31
3.16.2.3 cmdline . . . 32
3.16.2.4 elf_sec . . . 32
3.16.2.5 flags . . . 32
3.16.2.6 mem_lower . . . 32
3.16.2.7 mem_upper . . . 32
3.16.2.8 mmap_addr . . . 32
3.16.2.9 mmap_length . . . 32
3.16.2.10 mods_addr . . . 32
3.16.2.11 mods_count . . . 32
3.16.2.12 u . . . 32
3.17 pok_port_t Struct Reference . . . 32
3.17.1 Detailed Description . . . 33
3.17.2 Field Documentation . . . 33
3.17.2.1 direction . . . 33
3.17.2.2 discipline . . . 33
3.17.2.3 empty . . . 33
3.17.2.4 full . . . 33
3.17.2.5 identifier . . . 33
3.17.2.6 index . . . 33
3.17.2.7 kind . . . 34
3.17.2.8 last_receive . . . 34
3.17.2.9 lock . . . 34
3.17.2.10 must_be_flushed . . . 34
3.17.2.11 off_b . . . 34
3.17.2.12 off_e . . . 34
3.17.2.13 partition . . . 34
3.17.2.14 ready . . . 34
3.17.2.15 refresh . . . 34
3.17.2.16 size . . . 34

3.18 pok_space Struct Reference . . . 34
3.18.1 Detailed Description . . . 35
3.18.2 Field Documentation . . . 35
3.18.2.1 phys_base . . . 35
3.18.2.2 size . . . 35
3.19 pok_syscall_args_t Struct Reference . . . 35
3.19.1 Detailed Description . . . 35
3.19.2 Field Documentation . . . 35
3.19.2.1 arg1 . . . 35
3.19.2.2 arg2 . . . 35
3.19.2.3 arg3 . . . 36
3.19.2.4 arg4 . . . 36
3.19.2.5 arg5 . . . 36
3.19.2.6 nargs . . . 36
3.20 pok_syscall_info_t Struct Reference . . . 36
3.20.1 Detailed Description . . . 36
3.20.2 Field Documentation . . . 36
3.20.2.1 base_addr . . . 36
3.20.2.2 partition . . . 36
3.20.2.3 thread . . . 36
3.21 ppc_pte_t Struct Reference . . . 37
3.21.1 Detailed Description . . . 37
3.21.2 Field Documentation . . . 37
3.21.2.1 rpn_flags . . . 37
3.21.2.2 vsid_api . . . 37
3.22 space_context_t Struct Reference . . . 37
3.22.1 Detailed Description . . . 37
3.22.2 Field Documentation . . . 37
3.22.2.1 arg1 . . . 37
3.22.2.2 arg2 . . . 38
3.22.2.3 ctx . . . 38
3.22.2.4 fake_ret . . . 38
3.22.2.5 kernel_sp . . . 38
3.22.2.6 partition_id . . . 38
3.22.2.7 user_pc . . . 38
3.22.2.8 user_sp . . . 38
3.23 start_context_t Struct Reference . . . 38
3.23.1 Detailed Description . . . 38
3.23.2 Field Documentation . . . 39
3.23.2.1 ctx . . . 39

3.23.2.2 entry . . . 39
3.23.2.3 fake_ret . . . 39
3.23.2.4 id . . . 39
3.24 volatile_context_t Struct Reference . . . 39
3.24.1 Detailed Description . . . 40
3.24.2 Field Documentation . . . 40
3.24.2.1 back_chain . . . 40
3.24.2.2 cr . . . 40
3.24.2.3 ctr . . . 40
3.24.2.4 lr . . . 40
3.24.2.5 pad0 . . . 40
3.24.2.6 pad1 . . . 40
3.24.2.7 r0 . . . 40
3.24.2.8 r10 . . . 40
3.24.2.9 r11 . . . 40
3.24.2.10 r12 . . . 40
3.24.2.11 r13 . . . 40
3.24.2.12 r2 . . . 41
3.24.2.13 r3 . . . 41
3.24.2.14 r4 . . . 41
3.24.2.15 r5 . . . 41
3.24.2.16 r6 . . . 41
3.24.2.17 r7 . . . 41
3.24.2.18 r8 . . . 41
3.24.2.19 r9 . . . 41
3.24.2.20 sp . . . 41
3.24.2.21 srr0 . . . 41
3.24.2.22 srr1 . . . 41
3.24.2.23 unused_lr . . . 41
3.24.2.24 xer . . . 42

4 File Documentation **43**

4.1 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/arch.c File Reference . . . 43
4.1.1 Detailed Description . . . 43
4.1.2 Function Documentation . . . 43
4.1.2.1 pok_arch_event_register . . . 43
4.1.2.2 pok_arch_idle . . . 44
4.1.2.3 pok_arch_init . . . 44
4.1.2.4 pok_arch_preempt_disable . . . 44
4.1.2.5 pok_arch_preempt_enable . . . 44

4.1.2.6 pok_arch_space_init . . . 45
4.1.2.7 pok_thread_stack_addr . . . 45
4.2 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/arch.c File Reference . . . 45
4.2.1 Detailed Description . . . 46
4.2.2 Function Documentation . . . 46
4.2.2.1 pok_arch_event_register . . . 46
4.2.2.2 pok_arch_idle . . . 46
4.2.2.3 pok_arch_init . . . 46
4.2.2.4 pok_arch_preempt_disable . . . 47
4.2.2.5 pok_arch_preempt_enable . . . 47
4.2.2.6 pok_thread_stack_addr . . . 47
4.3 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/arch.c File Reference . . . 47
4.3.1 Detailed Description . . . 47
4.3.2 Function Documentation . . . 48
4.3.2.1 pok_arch_event_register . . . 48
4.3.2.2 pok_arch_idle . . . 48
4.3.2.3 pok_arch_init . . . 48
4.3.2.4 pok_arch_preempt_disable . . . 49
4.3.2.5 pok_arch_preempt_enable . . . 49
4.3.2.6 pok_thread_stack_addr . . . 49
4.4 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/msr.h File Reference . . . 49
4.4.1 Macro Definition Documentation . . . 49
4.4.1.1 MSR_DR . . . 49
4.4.1.2 MSR_EE . . . 50
4.4.1.3 MSR_IP . . . 50
4.4.1.4 MSR_IR . . . 50
4.4.1.5 MSR_PR . . . 50
4.5 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/bsp.c File Reference . . . 50
4.5.1 Function Documentation . . . 50
4.5.1.1 pok_bsp_init . . . 50
4.5.1.2 pok_bsp_mem_alloc . . . 50
4.5.2 Variable Documentation . . . 51
4.5.2.1 _end . . . 51
4.6 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/bsp.c File Reference . . . 51
4.6.1 Detailed Description . . . 51
4.6.2 Function Documentation . . . 51
4.6.2.1 pok_bsp_init . . . 51
4.6.2.2 pok_bsp_mem_alloc . . . 51
4.6.3 Variable Documentation . . . 52
4.6.3.1 _end . . . 52

4.7 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/bsp.c File Reference . . . 52
4.7.1 Function Documentation . . . 52
4.7.1.1 pok_bsp_init . . . 52
4.7.1.2 pok_bsp_irq_acknowledge . . . 52
4.7.1.3 pok_bsp_irq_register . . . 53
4.7.1.4 pok_bsp_mem_alloc . . . 53
4.7.1.5 pok_bsp_time_init . . . 53
4.8 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/cons.c File Reference . . . 53
4.8.1 Function Documentation . . . 53
4.8.1.1 pok_cons_init . . . 53
4.9 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/cons.c File Reference . . . 54
4.9.1 Detailed Description . . . 54
4.9.2 Function Documentation . . . 54
4.9.2.1 pok_cons_init . . . 54
4.10 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/cons.c File Reference . . . 54
4.10.1 Function Documentation . . . 54
4.10.1.1 pok_cons_init . . . 55
4.11 /home/phipse/gsoc/pok/trunk/kernel/core/cons.c File Reference . . . 55
4.12 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/cons.h File Reference . . . 55
4.12.1 Function Documentation . . . 55
4.12.1.1 pok_cons_init . . . 55
4.13 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/cons.h File Reference . . . 55
4.13.1 Detailed Description . . . 56
4.13.2 Macro Definition Documentation . . . 56
4.13.2.1 UART1 . . . 56
4.13.2.2 UART_CTRL_FL . . . 56
4.13.2.3 UART_CTRL_LB . . . 56
4.13.2.4 UART_CTRL_OFFSET . . . 56
4.13.2.5 UART_CTRL_PE . . . 56
4.13.2.6 UART_CTRL_PS . . . 56
4.13.2.7 UART_CTRL_RE . . . 56
4.13.2.8 UART_CTRL_RI . . . 57
4.13.2.9 UART_CTRL_TE . . . 57
4.13.2.10 UART_CTRL_TI . . . 57
4.13.2.11 UART_DATA_OFFSET . . . 57
4.13.2.12 UART_SCALER_OFFSET . . . 57
4.13.2.13 UART_STAT_OFFSET . . . 57
4.13.2.14 UART_STATUS_BR . . . 57
4.13.2.15 UART_STATUS_DR . . . 57
4.13.2.16 UART_STATUS_ERR . . . 57

4.13.2.17 UART_STATUS_FE . . . 58
4.13.2.18 UART_STATUS_OE . . . 58
4.13.2.19 UART_STATUS_PE . . . 58
4.13.2.20 UART_STATUS_THE . . . 58
4.13.2.21 UART_STATUS_TSE . . . 58
4.13.3 Function Documentation . . . 58
4.13.3.1 pok_cons_init . . . 58
4.14 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/cons.h File Reference . . . 58
4.14.1 Function Documentation . . . 58
4.14.1.1 pok_cons_init . . . 58
4.15 /home/phipse/gsoc/pok/trunk/kernel/include/core/cons.h File Reference . . . 59
4.16 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/ioports.h File Reference . . . 59
4.16.1 Macro Definition Documentation . . . 59
4.16.1.1 inb . . . 59
4.16.1.2 outb . . . 59
4.16.1.3 POK_PREP_IOBASE . . . 59
4.17 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/ioports.h File Reference . . . 59
4.17.1 Detailed Description . . . 59
4.18 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/ioports.h File Reference . . . 59
4.18.1 Macro Definition Documentation . . . 60
4.18.1.1 inb . . . 60
4.18.1.2 inl . . . 60
4.18.1.3 outb . . . 60
4.18.1.4 outl . . . 60
4.19 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/space.c File Reference . . . 61
4.19.1 Macro Definition Documentation . . . 62
4.19.1.1 KERNEL_STACK_SIZE . . . 62
4.19.1.2 POK_PAGE_MASK . . . 62
4.19.1.3 POK_PAGE_SIZE . . . 62
4.19.1.4 PPC_PTE_C . . . 62
4.19.1.5 PPC_PTE_G . . . 62
4.19.1.6 PPC_PTE_H . . . 62
4.19.1.7 PPC_PTE_I . . . 62
4.19.1.8 PPC_PTE_M . . . 62
4.19.1.9 PPC_PTE_PP_NO . . . 62
4.19.1.10 PPC_PTE_PP_RO . . . 62
4.19.1.11 PPC_PTE_PP_RW . . . 62
4.19.1.12 PPC_PTE_R . . . 62
4.19.1.13 PPC_PTE_V . . . 63
4.19.1.14 PPC_PTE_W . . . 63

4.19.1.15 PPC_SR_KP . . . 63
4.19.1.16 PPC_SR_Ks . . . 63
4.19.1.17 PPC_SR_T . . . 63
4.19.2 Function Documentation . . . 63
4.19.2.1 pok_arch_dsi_int . . . 63
4.19.2.2 pok_arch_isi_int . . . 63
4.19.2.3 pok_arch_rfi . . . 64
4.19.2.4 pok_arch_space_init . . . 64
4.19.2.5 pok_create_space . . . 64
4.19.2.6 pok_space_base_vaddr . . . 64
4.19.2.7 pok_space_context_create . . . 65
4.19.2.8 pok_space_switch . . . 65
4.19.3 Variable Documentation . . . 65
4.19.3.1 spaces . . . 65
4.20 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.c File Reference . . . 65
4.20.1 Detailed Description . . . 66
4.20.2 Macro Definition Documentation . . . 66
4.20.2.1 KERNEL_STACK_SIZE . . . 66
4.20.3 Function Documentation . . . 67
4.20.3.1 __attribute__ . . . 67
4.20.3.2 __attribute__ . . . 67
4.20.3.3 __attribute__ . . . 67
4.20.3.4 pok_arch_space_init . . . 67
4.20.3.5 pok_create_space . . . 68
4.20.3.6 pok_space_base_vaddr . . . 68
4.20.3.7 pok_space_context_create . . . 68
4.20.3.8 pok_space_switch . . . 69
4.20.4 Variable Documentation . . . 69
4.20.4.1 spaces . . . 69
4.21 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.c File Reference . . . 69
4.21.1 Detailed Description . . . 70
4.21.2 Macro Definition Documentation . . . 70
4.21.2.1 KERNEL_STACK_SIZE . . . 70
4.21.3 Function Documentation . . . 70
4.21.3.1 pok_create_space . . . 70
4.21.3.2 pok_dispatch_space . . . 70
4.21.3.3 pok_space_base_vaddr . . . 71
4.21.3.4 pok_space_context_create . . . 71
4.21.3.5 pok_space_switch . . . 72
4.22 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/syscalls.c File Reference . . . 72

4.22.1 Function Documentation . . . 72
4.22.1.1 pok_arch_sc_int . . . 72
4.23 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/syscalls.c File Reference . . . 73
4.23.1 Detailed Description . . . 73
4.23.2 Function Documentation . . . 73
4.23.2.1 pok_arch_sc_int . . . 73
4.23.2.2 pok_syscalls_init . . . 74
4.24 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/syscalls.c File Reference . . . 74
4.24.1 Detailed Description . . . 74
4.24.2 Macro Definition Documentation . . . 75
4.24.2.1 PARTITION_ID . . . 75
4.24.3 Function Documentation . . . 75
4.24.3.1 INTERRUPT_HANDLER_syscall . . . 75
4.24.3.2 pok_syscall_init . . . 75
4.25 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.c File Reference . . . 76
4.26 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/thread.c File Reference . . . 76
4.26.1 Detailed Description . . . 76
4.27 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.c File Reference . . . 76
4.28 /home/phipse/gsoc/pok/trunk/kernel/core/thread.c File Reference . . . 76
4.28.1 Detailed Description . . . 77
4.29 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.h File Reference . . . 77
4.29.1 Function Documentation . . . 77
4.29.1.1 pok_context_create . . . 77
4.29.1.2 pok_context_switch . . . 77
4.30 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/thread.h File Reference . . . 77
4.30.1 Detailed Description . . . 78
4.30.2 Function Documentation . . . 78
4.30.2.1 pok_context_create . . . 78
4.30.2.2 pok_context_switch . . . 78
4.30.3 Variable Documentation . . . 78
4.30.3.1 pok_arch_sp . . . 78
4.31 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.h File Reference . . . 78
4.31.1 Function Documentation . . . 78
4.31.1.1 pok_context_create . . . 78
4.31.1.2 pok_context_reset . . . 78
4.31.1.3 pok_context_switch . . . 78
4.32 /home/phipse/gsoc/pok/trunk/kernel/include/core/thread.h File Reference . . . 79
4.33 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/timer.c File Reference . . . 79
4.33.1 Macro Definition Documentation . . . 79
4.33.1.1 BUS_FREQ . . . 79

4.33.1.2 FREQ_DIV . . . 79
4.33.2 Function Documentation . . . 79
4.33.2.1 pok_arch_decr_int . . . 79
4.33.2.2 pok_bsp_time_init . . . 79
4.34 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/timer.c File Reference . . . 80
4.34.1 Detailed Description . . . 80
4.34.2 Function Documentation . . . 80
4.34.2.1 pok_bsp_time_init . . . 80
4.34.2.2 timer_isr . . . 81
4.35 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/context_offset.h File Reference . . . 81
4.35.1 Detailed Description . . . 82
4.35.2 Macro Definition Documentation . . . 82
4.35.2.1 G1_OFFSET . . . 82
4.35.2.2 G2_OFFSET . . . 82
4.35.2.3 G3_OFFSET . . . 82
4.35.2.4 G4_OFFSET . . . 82
4.35.2.5 G5_OFFSET . . . 82
4.35.2.6 G6_OFFSET . . . 82
4.35.2.7 G7_OFFSET . . . 82
4.35.2.8 I0_OFFSET . . . 82
4.35.2.9 I1_OFFSET . . . 82
4.35.2.10 I2_OFFSET . . . 82
4.35.2.11 I3_OFFSET . . . 83
4.35.2.12 I4_OFFSET . . . 83
4.35.2.13 I5_OFFSET . . . 83
4.35.2.14 I6_OFFSET . . . 83
4.35.2.15 I7_OFFSET . . . 83
4.35.2.16 L0_OFFSET . . . 83
4.35.2.17 L1_OFFSET . . . 83
4.35.2.18 L2_OFFSET . . . 83
4.35.2.19 L3_OFFSET . . . 83
4.35.2.20 L4_OFFSET . . . 83
4.35.2.21 L5_OFFSET . . . 83
4.35.2.22 L6_OFFSET . . . 83
4.35.2.23 L7_OFFSET . . . 84
4.35.2.24 NPC_OFFSET . . . 84
4.35.2.25 PC_OFFSET . . . 84
4.35.2.26 PSR_OFFSET . . . 84
4.35.2.27 RESTORE_CNT_OFFSET . . . 84
4.35.2.28 WIM_OFFSET . . . 84

4.35.2.29 Y_OFFSET . . . 84
4.36 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/irq.h File Reference . . . 84
4.36.1 Detailed Description . . . 84
4.36.2 Macro Definition Documentation . . . 85
4.36.2.1 ack_irq . . . 85
4.36.2.2 IRQMP_BASE . . . 85
4.36.2.3 IRQMP_CLEAR_OFFSET . . . 85
4.36.2.4 IRQMP_MASK0_OFFSET . . . 85
4.36.2.5 unmask_irq . . . 85
4.37 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/sparc_conf.h File Reference . . . 85
4.37.1 Detailed Description . . . 86
4.37.2 Macro Definition Documentation . . . 86
4.37.2.1 ASI_MMU_BYPASS . . . 86
4.37.2.2 SPARC_PAGE_SIZE . . . 86
4.37.2.3 SPARC_PARTITION_BASE_VADDR . . . 86
4.37.2.4 SPARC_PARTITION_SIZE . . . 86
4.37.2.5 SPARC_PROC_FREQ . . . 86
4.37.2.6 SPARC_RAM_ADDR . . . 86
4.37.2.7 WINDOWS_NBR . . . 86
4.38 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/timer.h File Reference . . . 87
4.38.1 Detailed Description . . . 87
4.38.2 Macro Definition Documentation . . . 87
4.38.2.1 TIMER1 . . . 87
4.38.2.2 TIMER_CNT_VAL_OFFSET . . . 87
4.38.2.3 TIMER_CTRL_CH . . . 87
4.38.2.4 TIMER_CTRL_DH . . . 87
4.38.2.5 TIMER_CTRL_EN . . . 88
4.38.2.6 TIMER_CTRL_IE . . . 88
4.38.2.7 TIMER_CTRL_IP . . . 88
4.38.2.8 TIMER_CTRL_LD . . . 88
4.38.2.9 TIMER_CTRL_OFFSET . . . 88
4.38.2.10 TIMER_CTRL_RS . . . 88
4.38.2.11 TIMER_IRQ . . . 88
4.38.2.12 TIMER_RELOAD_OFFSET . . . 88
4.38.2.13 TIMER_SCAL_RELOAD_OFFSET . . . 88
4.38.2.14 TIMER_SCALER_OFFSET . . . 89
4.39 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/psr.h File Reference . . . 89
4.39.1 Detailed Description . . . 89
4.39.2 Macro Definition Documentation . . . 89
4.39.2.1 PSR_CWP_MASK . . . 89

4.39.2.2 PSR_ET . . . 89
4.39.2.3 PSR_PIL . . . 89
4.39.2.4 PSR_PS . . . 89
4.39.2.5 PSR_S . . . 90
4.40 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.h File Reference . . . 90
4.40.1 Detailed Description . . . 91
4.40.2 Macro Definition Documentation . . . 91
4.40.2.1 ASI_M_MMUREGS . . . 91
4.40.2.2 LEON_CTX_NBR . . . 91
4.40.2.3 MM_ACC_E . . . 91
4.40.2.4 MM_ACC_R . . . 91
4.40.2.5 MM_ACC_R_S_RW . . . 91
4.40.2.6 MM_ACC_RE . . . 91
4.40.2.7 MM_ACC_RW . . . 92
4.40.2.8 MM_ACC_RWE . . . 92
4.40.2.9 MM_ACC_S_RE . . . 92
4.40.2.10 MM_ACC_S_RWE . . . 92
4.40.2.11 MM_CACHEABLE . . . 92
4.40.2.12 MM_ET_INVALID . . . 92
4.40.2.13 MM_ET_PTD . . . 92
4.40.2.14 MM_ET_PTE . . . 92
4.40.2.15 mm_index1 . . . 92
4.40.2.16 mm_index2 . . . 93
4.40.2.17 mm_index3 . . . 93
4.40.2.18 MM_LVL1_ENTRIES_NBR . . . 93
4.40.2.19 MM_LVL1_PAGE_SIZE . . . 93
4.40.2.20 MM_LVL2_ENTRIES_NBR . . . 93
4.40.2.21 MM_LVL2_PAGE_SIZE . . . 93
4.40.2.22 MM_LVL3_ENTRIES_NBR . . . 93
4.40.2.23 MM_LVL3_PAGE_SIZE . . . 93
4.40.2.24 MM_MODIFIED . . . 93
4.40.2.25 MM_REFERENCED . . . 93
4.40.2.26 MMU_CTRL_REG . . . 94
4.40.2.27 MMU_CTX_REG . . . 94
4.40.2.28 MMU_CTXTBL_PTR . . . 94
4.40.2.29 MMU_FAULT_ADDR . . . 94
4.40.2.30 MMU_FAULT_STATUS . . . 94
4.40.3 Typedef Documentation . . . 94
4.40.3.1 ptd . . . 94
4.40.3.2 pte . . . 94

4.40.4 Function Documentation . . . 94
4.40.4.1 pok_arch_space_init . . . 94
4.41 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.h File Reference . . . 94
4.41.1 Detailed Description . . . 95
4.42 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/syscalls.h File Reference . . . 95
4.42.1 Detailed Description . . . 95
4.42.2 Function Documentation . . . 95
4.42.2.1 pok_syscalls_init . . . 95
4.43 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/traps.c File Reference . . . 95
4.43.1 Detailed Description . . . 96
4.43.2 Function Documentation . . . 96
4.43.2.1 trap_handler . . . 96
4.43.2.2 traps_init . . . 97
4.43.3 Variable Documentation . . . 97
4.43.3.1 pok_sparc_isr . . . 97
4.44 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/traps.h File Reference . . . 97
4.44.1 Detailed Description . . . 97
4.44.2 Macro Definition Documentation . . . 98
4.44.2.1 SPARC_TRAP_IRQ_BASE . . . 98
4.44.2.2 SPARC_TRAP_SYSCALL_BASE . . . 98
4.44.3 Typedef Documentation . . . 98
4.44.3.1 sparc_traps_handler . . . 98
4.44.4 Function Documentation . . . 98
4.44.4.1 traps_init . . . 98
4.44.5 Variable Documentation . . . 98
4.44.5.1 pok_sparc_isr . . . 98
4.45 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.c File Reference . . . 98
4.45.1 Macro Definition Documentation . . . 99
4.45.1.1 IDT_SIZE . . . 99
4.45.2 Function Documentation . . . 99
4.45.2.1 pok_event_init . . . 99
4.45.2.2 pok_idt_init . . . 99
4.45.2.3 pok_idt_set_gate . . . 99
4.45.3 Variable Documentation . . . 100
4.45.3.1 pok_idt . . . 100
4.46 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.h File Reference . . . 100
4.46.1 Macro Definition Documentation . . . 101
4.46.1.1 EXCEPTION_ALIGNEMENT_CHECK . . . 101
4.46.1.2 EXCEPTION_BOUNDRANGE . . . 101
4.46.1.3 EXCEPTION_BREAKPOINT . . . 101

4.46.1.4 EXCEPTION_COPSEG_OVERRUN . . . 101
4.46.1.5 EXCEPTION_DEBUG . . . 101
4.46.1.6 EXCEPTION_DIVIDE_ERROR . . . 101
4.46.1.7 EXCEPTION_DOUBLEFAULT . . . 101
4.46.1.8 EXCEPTION_FPU_FAULT . . . 101
4.46.1.9 EXCEPTION_GENERAL_PROTECTION . . . 101
4.46.1.10 EXCEPTION_INVALID_TSS . . . 102
4.46.1.11 EXCEPTION_INVALIDOPCODE . . . 102
4.46.1.12 EXCEPTION_MACHINE_CHECK . . . 102
4.46.1.13 EXCEPTION_NMI . . . 102
4.46.1.14 EXCEPTION_NOMATH_COPROC . . . 102
4.46.1.15 EXCEPTION_OVERFLOW . . . 102
4.46.1.16 EXCEPTION_PAGEFAULT . . . 102
4.46.1.17 EXCEPTION_RESERVED . . . 102
4.46.1.18 EXCEPTION_SEGMENT_NOT_PRESENT . . . 102
4.46.1.19 EXCEPTION_SIMD_FAULT . . . 102
4.46.1.20 EXCEPTION_STACKSEG_FAULT . . . 102
4.46.2 Typedef Documentation . . . 102
4.46.2.1 e_idte_type . . . 102
4.46.3 Enumeration Type Documentation . . . 103
4.46.3.1 e_idte_type . . . 103
4.46.4 Function Documentation . . . 103
4.46.4.1 pok_event_init . . . 103
4.46.4.2 pok_exception_init . . . 103
4.46.4.3 pok_idt_init . . . 103
4.46.4.4 pok_idt_set_gate . . . 103
4.46.4.5 pok_syscall_init . . . 104
4.47 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/exceptions.c File Reference . . . 104
4.47.1 Detailed Description . . . 104
4.48 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.c File Reference . . . 104
4.48.1 Macro Definition Documentation . . . 105
4.48.1.1 GDT_SIZE . . . 105
4.48.1.2 POK_CONFIG_NB_PARTITIONS . . . 105
4.48.1.3 POK_CONFIG_NB_THREADS . . . 105
4.48.2 Function Documentation . . . 105
4.48.2.1 gdt_disable . . . 105
4.48.2.2 gdt_enable . . . 105
4.48.2.3 gdt_set_segment . . . 105
4.48.2.4 gdt_set_system . . . 106
4.48.2.5 pok_gdt_init . . . 106

4.48.2.6 pok_tss_init . . . 107
4.48.2.7 tss_set_esp0 . . . 107
4.48.3 Variable Documentation . . . 107
4.48.3.1 pok_gdt . . . 107
4.48.3.2 pok_tss . . . 107
4.49 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.h File Reference . . . 107
4.49.1 Macro Definition Documentation . . . 108
4.49.1.1 GDT_BUILD_SELECTOR . . . 108
4.49.1.2 GDT_CORE_CODE_SEGMENT . . . 108
4.49.1.3 GDT_CORE_DATA_SEGMENT . . . 108
4.49.1.4 GDT_PARTITION_CODE_SEGMENT . . . 108
4.49.1.5 GDT_PARTITION_DATA_SEGMENT . . . 108
4.49.1.6 GDT_TSS_SEGMENT . . . 108
4.49.2 Typedef Documentation . . . 109
4.49.2.1 e_gdte_type . . . 109
4.49.3 Enumeration Type Documentation . . . 109
4.49.3.1 e_gdte_type . . . 109
4.49.4 Function Documentation . . . 109
4.49.4.1 gdt_disable . . . 109
4.49.4.2 gdt_enable . . . 109
4.49.4.3 gdt_set_segment . . . 109
4.49.4.4 gdt_set_system . . . 110
4.49.4.5 pok_gdt_init . . . 110
4.49.4.6 pok_tss_init . . . 110
4.49.4.7 tss_set_esp0 . . . 111
4.50 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/interrupt.c File Reference . . . 111
4.50.1 Function Documentation . . . 111
4.50.1.1 update_tss . . . 111
4.51 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/pci.c File Reference . . . 111
4.52 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/sysdesc.h File Reference . . . 111
4.53 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/tss.h File Reference . . . 112
4.54 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/types.h File Reference . . . 112
4.54.1 Macro Definition Documentation . . . 112
4.54.1.1 __POK_X86_TYPES_H__ . . . 112
4.54.2 Typedef Documentation . . . 112
4.54.2.1 int16_t . . . 112
4.54.2.2 int64_t . . . 112
4.54.2.3 int8_t . . . 112
4.54.2.4 intptr_t . . . 112
4.54.2.5 size_t . . . 113

4.54.2.6 uint16_t . . . 113
4.54.2.7 uint32_t . . . 113
4.54.2.8 uint64_t . . . 113
4.54.2.9 uint8_t . . . 113
4.55 /home/phipse/gsoc/pok/trunk/kernel/include/arch/sparc/types.h File Reference . . . 113
4.55.1 Typedef Documentation . . . 113
4.55.1.1 int16_t . . . 113
4.55.1.2 int64_t . . . 113
4.55.1.3 int8_t . . . 113
4.55.1.4 intptr_t . . . 114
4.55.1.5 size_t . . . 114
4.55.1.6 uint16_t . . . 114
4.55.1.7 uint32_t . . . 114
4.55.1.8 uint64_t . . . 114
4.55.1.9 uint8_t . . . 114
4.56 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/types.h File Reference . . . 114
4.56.1 Macro Definition Documentation . . . 114
4.56.1.1 __POK_X86_TYPES_H__ . . . 114
4.56.2 Typedef Documentation . . . 115
4.56.2.1 int16_t . . . 115
4.56.2.2 int64_t . . . 115
4.56.2.3 int8_t . . . 115
4.56.2.4 intptr_t . . . 115
4.56.2.5 size_t . . . 115
4.56.2.6 uint16_t . . . 115
4.56.2.7 uint32_t . . . 115
4.56.2.8 uint64_t . . . 115
4.56.2.9 uint8_t . . . 115
4.57 /home/phipse/gsoc/pok/trunk/kernel/include/types.h File Reference . . . 115
4.57.1 Macro Definition Documentation . . . 116
4.57.1.1 bool_t . . . 116
4.57.1.2 FALSE . . . 116
4.57.1.3 NULL . . . 116
4.57.1.4 pok_bool_t . . . 116
4.57.1.5 TRUE . . . 116
4.57.2 Typedef Documentation . . . 116
4.57.2.1 pok_blackboard_id_t . . . 116
4.57.2.2 pok_buffer_id_t . . . 116
4.57.2.3 pok_event_id_t . . . 117
4.57.2.4 pok_lockobj_id_t . . . 117

4.57.2.5 pok_partition_id_t . . . 117
4.57.2.6 pok_port_direction_t . . . 117
4.57.2.7 pok_port_id_t . . . 117
4.57.2.8 pok_port_kind_t . . . 117
4.57.2.9 pok_port_size_t . . . 117
4.57.2.10 pok_queueing_discipline_t . . . 117
4.57.2.11 pok_range_t . . . 117
4.57.2.12 pok_sem_id_t . . . 117
4.57.2.13 pok_sem_value_t . . . 117
4.57.2.14 pok_size_t . . . 117
4.58 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/debug.c File Reference . . . 118
4.59 /home/phipse/gsoc/pok/trunk/kernel/core/debug.c File Reference . . . 118
4.60 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pic.c File Reference . . . 118
4.60.1 Function Documentation . . . 118
4.60.1.1 pok_pic_eoi . . . 118
4.60.1.2 pok_pic_init . . . 118
4.60.1.3 pok_pic_mask . . . 119
4.60.1.4 pok_pic_unmask . . . 119
4.61 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pic.h File Reference . . . 119
4.61.1 Macro Definition Documentation . . . 120
4.61.1.1 PIC_MASTER_BASE . . . 120
4.61.1.2 PIC_MASTER_ICW1 . . . 120
4.61.1.3 PIC_MASTER_ICW2 . . . 120
4.61.1.4 PIC_MASTER_ICW3 . . . 120
4.61.1.5 PIC_MASTER_ICW4 . . . 120
4.61.1.6 PIC_SLAVE_BASE . . . 120
4.61.1.7 PIC_SLAVE_ICW1 . . . 120
4.61.1.8 PIC_SLAVE_ICW2 . . . 120
4.61.1.9 PIC_SLAVE_ICW3 . . . 120
4.61.1.10 PIC_SLAVE_ICW4 . . . 120
4.61.2 Function Documentation . . . 121
4.61.2.1 pok_pic_eoi . . . 121
4.61.2.2 pok_pic_init . . . 121
4.61.2.3 pok_pic_mask . . . 121
4.61.2.4 pok_pic_unmask . . . 121
4.62 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pit.c File Reference . . . 122
4.62.1 Macro Definition Documentation . . . 122
4.62.1.1 OSCILLATOR_RATE . . . 122
4.62.1.2 PIT_BASE . . . 122
4.62.1.3 PIT_IRQ . . . 122

4.62.2 Function Documentation . . . 123
4.62.2.1 INTERRUPT_HANDLER . . . 123
4.62.2.2 pok_x86_qemu_timer_init . . . 123
4.63 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pit.h File Reference . . . 123
4.63.1 Function Documentation . . . 123
4.63.1.1 pok_x86_qemu_timer_init . . . 123
4.64 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pm.c File Reference . . . 123
4.64.1 Detailed Description . . . 124
4.64.2 Macro Definition Documentation . . . 124
4.64.2.1 ALIGN_UP . . . 124
4.64.3 Function Documentation . . . 124
4.64.3.1 pok_pm_init . . . 124
4.64.3.2 pok_pm_sbrk . . . 125
4.64.4 Variable Documentation . . . 125
4.64.4.1 __pok_begin . . . 125
4.64.4.2 __pok_end . . . 125
4.64.4.3 pok_multiboot_info . . . 125
4.64.4.4 pok_multiboot_magic . . . 125
4.64.4.5 pok_x86_pm_brk . . . 125
4.64.4.6 pok_x86_pm_heap_end . . . 125
4.64.4.7 pok_x86_pm_heap_start . . . 125
4.65 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pm.h File Reference . . . 125
4.65.1 Macro Definition Documentation . . . 126
4.65.1.1 MEM_16MB . . . 126
4.65.2 Function Documentation . . . 126
4.65.2.1 pok_pm_init . . . 126
4.65.2.2 pok_pm_sbrk . . . 126
4.66 /home/phipse/gsoc/pok/trunk/kernel/core/boot.c File Reference . . . 126
4.66.1 Detailed Description . . . 127
4.66.2 Function Documentation . . . 127
4.66.2.1 pok_boot . . . 127
4.67 /home/phipse/gsoc/pok/trunk/kernel/core/error.c File Reference . . . 128
4.68 /home/phipse/gsoc/pok/trunk/kernel/core/instrumentation.c File Reference . . . 128
4.69 /home/phipse/gsoc/pok/trunk/kernel/core/kernel.c File Reference . . . 128
4.70 /home/phipse/gsoc/pok/trunk/kernel/core/loader.c File Reference . . . 128
4.70.1 Detailed Description . . . 128
4.71 /home/phipse/gsoc/pok/trunk/kernel/core/lockobj.c File Reference . . . 128
4.71.1 Detailed Description . . . 128
4.72 /home/phipse/gsoc/pok/trunk/kernel/core/partition.c File Reference . . . 128
4.72.1 Detailed Description . . . 129

4.73 /home/phipse/gsoc/pok/trunk/kernel/core/sched.c File Reference . . . 129
4.74 /home/phipse/gsoc/pok/trunk/kernel/core/syscall.c File Reference . . . 129
4.74.1 Function Documentation . . . 129
4.74.1.1 pok_core_syscall . . . 129
4.75 /home/phipse/gsoc/pok/trunk/kernel/core/time.c File Reference . . . 134
4.75.1 Detailed Description . . . 134
4.76 /home/phipse/gsoc/pok/trunk/kernel/include/arch.h File Reference . . . 134
4.76.1 Detailed Description . . . 135
4.76.2 Function Documentation . . . 135
4.76.2.1 pok_arch_event_register . . . 135
4.76.2.2 pok_arch_idle . . . 135
4.76.2.3 pok_arch_init . . . 136
4.76.2.4 pok_arch_preempt_disable . . . 136
4.76.2.5 pok_arch_preempt_enable . . . 136
4.76.2.6 pok_context_create . . . 136
4.76.2.7 pok_context_reset . . . 136
4.76.2.8 pok_context_switch . . . 136
4.76.2.9 pok_create_space . . . 136
4.76.2.10 pok_dispatch_space . . . 137
4.76.2.11 pok_space_base_vaddr . . . 137
4.76.2.12 pok_space_context_create . . . 138
4.76.2.13 pok_space_context_restart . . . 138
4.76.2.14 pok_space_switch . . . 138
4.76.2.15 pok_thread_stack_addr . . . 138
4.77 /home/phipse/gsoc/pok/trunk/kernel/include/arch/ppc/spinlock.h File Reference . . . 139
4.77.1 Macro Definition Documentation . . . 139
4.77.1.1 SPIN_LOCK . . . 139
4.77.1.2 SPIN_UNLOCK . . . 139
4.77.2 Typedef Documentation . . . 139
4.77.2.1 pok_spinlock_t . . . 139
4.78 /home/phipse/gsoc/pok/trunk/kernel/include/arch/sparc/spinlock.h File Reference . . . 140
4.78.1 Macro Definition Documentation . . . 140
4.78.1.1 SPIN_LOCK . . . 140
4.78.1.2 SPIN_UNLOCK . . . 140
4.78.2 Typedef Documentation . . . 140
4.78.2.1 pok_spinlock_t . . . 140
4.79 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/spinlock.h File Reference . . . 140
4.79.1 Macro Definition Documentation . . . 141
4.79.1.1 SPIN_LOCK . . . 141
4.79.1.2 SPIN_UNLOCK . . . 141

4.79.2 Typedef Documentation . . . 141
4.79.2.1 pok_spinlock_t . . . 141
4.80 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/interrupt.h File Reference . . . 141
4.80.1 Macro Definition Documentation . . . 142
4.80.1.1 INTERRUPT_HANDLER . . . 142
4.80.1.2 INTERRUPT_HANDLER_errorcode . . . 142
4.80.1.3 INTERRUPT_HANDLER_syscall . . . 142
4.80.2 Function Documentation . . . 143
4.80.2.1 update_tss . . . 143
4.80.3 Variable Documentation . . . 143
4.80.3.1 pok_tss . . . 143
4.81 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h File Reference . . . 143
4.81.1 Detailed Description . . . 144
4.81.2 Macro Definition Documentation . . . 144
4.81.2.1 EXT_C . . . 144
4.81.2.2 MULTIBOOT_BOOTLOADER_MAGIC . . . 144
4.81.2.3 MULTIBOOT_BOOTLOADER_MAGIC . . . 144
4.81.2.4 MULTIBOOT_CMDLINE . . . 144
4.81.2.5 MULTIBOOT_HEADER_FLAGS . . . 144
4.81.2.6 MULTIBOOT_HEADER_MAGIC . . . 144
4.81.2.7 MULTIBOOT_MODS . . . 145
4.81.2.8 MULTIBOOT_STACK_SIZE . . . 145
4.82 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/pci.h File Reference . . . 145
4.83 /home/phipse/gsoc/pok/trunk/kernel/include/bsp.h File Reference . . . 145
4.83.1 Detailed Description . . . 145
4.83.2 Function Documentation . . . 145
4.83.2.1 pok_bsp_init . . . 145
4.83.2.2 pok_bsp_irq_acknowledge . . . 146
4.83.2.3 pok_bsp_irq_register . . . 146
4.83.2.4 pok_bsp_mem_alloc . . . 146
4.83.2.5 pok_bsp_time_init . . . 146
4.83.2.6 pok_cons_write . . . 147
4.84 /home/phipse/gsoc/pok/trunk/kernel/include/core/boot.h File Reference . . . 147
4.84.1 Detailed Description . . . 147
4.84.2 Function Documentation . . . 147
4.84.2.1 pok_boot . . . 147
4.85 /home/phipse/gsoc/pok/trunk/kernel/include/core/cpio.h File Reference . . . 148
4.85.1 Enumeration Type Documentation . . . 148
4.85.1.1 cpio_format . . . 148
4.85.2 Function Documentation . . . 149

4.85.2.1 cpio_get_fileaddr . . . 149
4.85.2.2 cpio_get_filename . . . 149
4.85.2.3 cpio_next_file . . . 149
4.85.2.4 cpio_open . . . 149
4.86 /home/phipse/gsoc/pok/trunk/kernel/include/core/debug.h File Reference . . . 149
4.86.1 Macro Definition Documentation . . . 149
4.86.1.1 POK_DEBUG_PRINT_CURRENT_STATE . . . 149
4.86.1.2 POK_FATAL . . . 149
4.87 /home/phipse/gsoc/pok/trunk/kernel/include/core/error.h File Reference . . . 149
4.88 /home/phipse/gsoc/pok/trunk/kernel/include/core/instrumentation.h File Reference . . . 149
4.89 /home/phipse/gsoc/pok/trunk/kernel/include/core/kernel.h File Reference . . . 149
4.89.1 Function Documentation . . . 149
4.89.1.1 pok_kernel_restart . . . 149
4.89.1.2 pok_kernel_stop . . . 149
4.90 /home/phipse/gsoc/pok/trunk/kernel/include/core/loader.h File Reference . . . 150
4.90.1 Function Documentation . . . 150
4.90.1.1 pok_loader_load_partition . . . 150
4.91 /home/phipse/gsoc/pok/trunk/kernel/include/core/lockobj.h File Reference . . . 150
4.91.1 Macro Definition Documentation . . . 151
4.91.1.1 POK_CONFIG_NB_LOCKOBJECTS . . . 151
4.91.2 Enumeration Type Documentation . . . 151
4.91.2.1 pok_locking_policy_t . . . 151
4.91.2.2 pok_lockobj_kind_t . . . 151
4.91.2.3 pok_lockobj_lock_kind_t . . . 152
4.91.2.4 pok_lockobj_operation_t . . . 152
4.91.2.5 pok_mutex_state_t . . . 152
4.91.3 Function Documentation . . . 152
4.91.3.1 pok_lockobj_create . . . 152
4.91.3.2 pok_lockobj_eventbroadcast . . . 152
4.91.3.3 pok_lockobj_eventsignal . . . 152
4.91.3.4 pok_lockobj_eventwait . . . 153
4.91.3.5 pok_lockobj_init . . . 153
4.91.3.6 pok_lockobj_lock . . . 153
4.91.3.7 pok_lockobj_partition_create . . . 153
4.91.3.8 pok_lockobj_partition_wrapper . . . 153
4.91.3.9 pok_lockobj_unlock . . . 153
4.92 /home/phipse/gsoc/pok/trunk/kernel/include/core/partition.h File Reference . . . 153
4.92.1 Detailed Description . . . 153
4.93 /home/phipse/gsoc/pok/trunk/kernel/include/core/sched.h File Reference . . . 153
4.94 /home/phipse/gsoc/pok/trunk/kernel/include/core/schedvalues.h File Reference . . . 153

4.94.1 Enumeration Type Documentation . . . 153
4.94.1.1 pok_sched_t . . . 153
4.95 /home/phipse/gsoc/pok/trunk/kernel/include/core/syscall.h File Reference . . . 154
4.95.1 Macro Definition Documentation . . . 154
4.95.1.1 POK_CHECK_PTR_OR_RETURN . . . 154
4.95.2 Enumeration Type Documentation . . . 155
4.95.2.1 pok_syscall_id_t . . . 155
4.95.3 Function Documentation . . . 156
4.95.3.1 pok_core_syscall . . . 156
4.95.3.2 pok_syscall_init . . . 160
4.96 /home/phipse/gsoc/pok/trunk/kernel/include/core/time.h File Reference . . . 161
4.97 /home/phipse/gsoc/pok/trunk/kernel/include/dependencies.h File Reference . . . 161
4.98 /home/phipse/gsoc/pok/trunk/kernel/include/elf.h File Reference . . . 161
4.98.1 Macro Definition Documentation . . . 161
4.98.1.1 EI_NIDENT . . . 161
4.98.2 Typedef Documentation . . . 161
4.98.2.1 Elf32_Addr . . . 161
4.98.2.2 Elf32_Half . . . 161
4.98.2.3 Elf32_Off . . . 161
4.98.2.4 Elf32_Word . . . 162
4.99 /home/phipse/gsoc/pok/trunk/kernel/include/errno.h File Reference . . . 162
4.99.1 Enumeration Type Documentation . . . 162
4.99.1.1 pok_ret_t . . . 162
4.100 /home/phipse/gsoc/pok/trunk/kernel/include/libc.h File Reference . . . 163
4.100.1 Function Documentation . . . 164
4.100.1.1 memcpy . . . 164
4.100.1.2 memset . . . 164
4.100.1.3 strcmp . . . 164
4.100.1.4 strlen . . . 164
4.100.1.5 strncmp . . . 164
4.101 /home/phipse/gsoc/pok/trunk/kernel/include/middleware/port.h File Reference . . . 164
4.101.1 Detailed Description . . . 165
4.101.2 Macro Definition Documentation . . . 165
4.101.2.1 POK_PORT_MAX_SIZE . . . 165
4.101.3 Typedef Documentation . . . 165
4.101.3.1 pok_port_queueing_discipline_t . . . 165
4.101.4 Enumeration Type Documentation . . . 165
4.101.4.1 pok_port_directions_t . . . 165
4.101.4.2 pok_port_kinds_t . . . 166
4.101.4.3 pok_port_queueing_disciplines_t . . . 166

4.102/home/phipse/gsoc/pok/trunk/kernel/include/middleware/queue.h File Reference 166
4.103/home/phipse/gsoc/pok/trunk/kernel/libc/__udivdi3.c File Reference 166
4.103.1 Function Documentation 166
4.103.1.1 __udivdi3 166
4.104/home/phipse/gsoc/pok/trunk/kernel/libc/memcpy.c File Reference 167
4.104.1 Function Documentation 167
4.104.1.1 memcpy 167
4.105/home/phipse/gsoc/pok/trunk/kernel/libc/memset.c File Reference 168
4.105.1 Function Documentation 168
4.105.1.1 __attribute__ 168
4.106/home/phipse/gsoc/pok/trunk/kernel/libc/printf.c File Reference 168
4.107/home/phipse/gsoc/pok/trunk/kernel/libc/strcmp.c File Reference 168
4.108/home/phipse/gsoc/pok/trunk/kernel/libc/strlen.c File Reference 168
4.109/home/phipse/gsoc/pok/trunk/kernel/middleware/portcreate.c File Reference 168
4.110/home/phipse/gsoc/pok/trunk/kernel/middleware/portflushall.c File Reference 168
4.110.1 Detailed Description 168
4.111/home/phipse/gsoc/pok/trunk/kernel/middleware/portinit.c File Reference 169
4.112/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingcreate.c File Reference 169
4.113/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingid.c File Reference 169
4.114/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingreceive.c File Reference 169
4.115/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingsend.c File Reference 169
4.116/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingstatus.c File Reference 169
4.117/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingcreate.c File Reference 169
4.118/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingid.c File Reference 169
4.119/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingread.c File Reference 169
4.120/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingstatus.c File Reference 169
4.121/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingwrite.c File Reference 169
4.121.1 Detailed Description 169
4.122/home/phipse/gsoc/pok/trunk/kernel/middleware/portutils.c File Reference 170
4.122.1 Detailed Description 170
4.123/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualdestination.c File Reference 170
4.124/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualgetglobal.c File Reference 170
4.125/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualid.c File Reference 170
4.126/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualnbdestinations.c File Reference 170
4.127/home/phipse/gsoc/pok/trunk/kernel/middleware/queueinit.c File Reference 170
4.128/home/phipse/gsoc/pok/trunk/kernel/middleware/ressources.c File Reference 170

Index **170**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

__attribute__ . . . 7
context_t . . . 12
cpio_bin_header . . . 16
cpio_file . . . 18
Elf32_Ehdr . . . 19
Elf32_Phdr . . . 20
interrupt_frame . . . 22
pok_aout_symbol_table_t . . . 24
pok_elf_section_header_table_t . . . 24
pok_lockobj_attr_t . . . 25
pok_lockobj_lockattr_t . . . 26
pok_lockobj_t . . . 27
pok_memory_map_t . . . 28
pok_module_t . . . 29
pok_multiboot_header_t . . . 30
pok_multiboot_info_t . . . 31
pok_port_t . . . 32
pok_space . . . 34
pok_syscall_args_t . . . 35
pok_syscall_info_t . . . 36
ppc_pte_t . . . 37
space_context_t . . . 37
start_context_t . . . 38
volatile_context_t . . . 39

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/arch.c
Provide generic architecture access for PPC architecture . . . 43
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/msr.h . . . 49
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/space.c . . . 61
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/syscalls.c . . . 72
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.c . . . 76
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.h . . . 77
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/timer.c . . . 79
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/bsp.c . . . 50
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/cons.c . . . 53
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/cons.h . . . 55
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/ioports.h . . . 59
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/arch.c . . . 45
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/context_offset.h
Define registers offset in context stack . . . 81
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/psr.h
Processor State Register utils . . . 89
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.c
Memory management in SPARC . . . 65
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.h . . . 90
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/syscalls.c
Syscalls management in SPARC . . . 73
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/syscalls.h . . . 95
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/thread.c
Thread management . . . 76
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/thread.h . . . 77
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/traps.c
Traps management . . . 95
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/traps.h . . . 97
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/bsp.c . . . 51
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/cons.c
Leon3 UART driver . . . 54
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/cons.h . . . 55
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/ioports.h
SPARC "ioports". Use MMU bypass to access IO memory . . . 59
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/irq.h
Leon3 IRQ management . . . 84

/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/sparc_conf.h
Define all constant values for a SPARC bsp . . . 85
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/timer.c
Leon3 timer management . . . 80
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/timer.h . . . 87
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/arch.c
Provides generic architecture interface for x86 architecture . . . 47
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.c . . . 98
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.h . . . 100
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/exceptions.c . . . 104
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.c . . . 104
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.h . . . 107
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/interrupt.c . . . 111
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/pci.c . . . 111
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.c
Handle address spaces . . . 69
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.h . . . 94
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/syscalls.c
This file implement system-calls for x86 platform . . . 74
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/sysdesc.h . . . 111
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.c . . . 76
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.h . . . 78
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/tss.h . . . 112
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/types.h . . . 112
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/bsp.c . . . 52
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/cons.c . . . 54
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/cons.h . . . 58
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/debug.c . . . 118
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pic.c . . . 118
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pic.h . . . 119
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pit.c . . . 122
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pit.h . . . 123
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pm.c . . . 123
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pm.h . . . 125
/home/phipse/gsoc/pok/trunk/kernel/core/boot.c
Boot function to start the kernel . . . 126
/home/phipse/gsoc/pok/trunk/kernel/core/cons.c . . . 55
/home/phipse/gsoc/pok/trunk/kernel/core/debug.c . . . 118
/home/phipse/gsoc/pok/trunk/kernel/core/error.c . . . 128
/home/phipse/gsoc/pok/trunk/kernel/core/instrumentation.c . . . 128
/home/phipse/gsoc/pok/trunk/kernel/core/kernel.c . . . 128
/home/phipse/gsoc/pok/trunk/kernel/core/loader.c . . . 128
/home/phipse/gsoc/pok/trunk/kernel/core/lockobj.c
Provides functionnalities for locking functions (mutexes, semaphores and so on) . . . 128
/home/phipse/gsoc/pok/trunk/kernel/core/partition.c
This file provides functions for partitioning services . . . 128
/home/phipse/gsoc/pok/trunk/kernel/core/sched.c . . . 129
/home/phipse/gsoc/pok/trunk/kernel/core/syscall.c . . . 129
/home/phipse/gsoc/pok/trunk/kernel/core/thread.c
Thread management in kernel . . . 76
/home/phipse/gsoc/pok/trunk/kernel/core/time.c . . . 134
/home/phipse/gsoc/pok/trunk/kernel/include/arch.h
Generic interface to handle architectures . . . 134
/home/phipse/gsoc/pok/trunk/kernel/include/bsp.h
Interfaces that BSP must provide . . . 145
/home/phipse/gsoc/pok/trunk/kernel/include/dependencies.h . . . 161
/home/phipse/gsoc/pok/trunk/kernel/include/elf.h . . . 161
/home/phipse/gsoc/pok/trunk/kernel/include/errno.h . . . 162

/home/phipse/gsoc/pok/trunk/kernel/include/libc.h . . . 163
/home/phipse/gsoc/pok/trunk/kernel/include/types.h . . . 115
/home/phipse/gsoc/pok/trunk/kernel/include/arch/ppc/spinlock.h . . . 139
/home/phipse/gsoc/pok/trunk/kernel/include/arch/sparc/spinlock.h . . . 140
/home/phipse/gsoc/pok/trunk/kernel/include/arch/sparc/types.h . . . 113
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/interrupt.h . . . 141
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/ioports.h . . . 59
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h . . . 143
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/pci.h . . . 145
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/spinlock.h . . . 140
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/types.h . . . 114
/home/phipse/gsoc/pok/trunk/kernel/include/core/boot.h . . . 147
/home/phipse/gsoc/pok/trunk/kernel/include/core/cons.h . . . 59
/home/phipse/gsoc/pok/trunk/kernel/include/core/cpio.h . . . 148
/home/phipse/gsoc/pok/trunk/kernel/include/core/debug.h . . . 149
/home/phipse/gsoc/pok/trunk/kernel/include/core/error.h . . . 149
/home/phipse/gsoc/pok/trunk/kernel/include/core/instrumentation.h . . . 149
/home/phipse/gsoc/pok/trunk/kernel/include/core/kernel.h . . . 149
/home/phipse/gsoc/pok/trunk/kernel/include/core/loader.h . . . 150
/home/phipse/gsoc/pok/trunk/kernel/include/core/lockobj.h . . . 150
/home/phipse/gsoc/pok/trunk/kernel/include/core/partition.h
Definition of structure for partitioning services . . . 153
/home/phipse/gsoc/pok/trunk/kernel/include/core/sched.h . . . 153
/home/phipse/gsoc/pok/trunk/kernel/include/core/schedvalues.h . . . 153
/home/phipse/gsoc/pok/trunk/kernel/include/core/syscall.h . . . 154
/home/phipse/gsoc/pok/trunk/kernel/include/core/thread.h . . . 79
/home/phipse/gsoc/pok/trunk/kernel/include/core/time.h . . . 161
/home/phipse/gsoc/pok/trunk/kernel/include/middleware/port.h
Describe queueing and sampling ports structures . . . 164
/home/phipse/gsoc/pok/trunk/kernel/include/middleware/queue.h . . . 166
/home/phipse/gsoc/pok/trunk/kernel/libc/__udivdi3.c . . . 166
/home/phipse/gsoc/pok/trunk/kernel/libc/memcpy.c . . . 167
/home/phipse/gsoc/pok/trunk/kernel/libc/memset.c . . . 168
/home/phipse/gsoc/pok/trunk/kernel/libc/printf.c . . . 168
/home/phipse/gsoc/pok/trunk/kernel/libc/strcmp.c . . . 168
/home/phipse/gsoc/pok/trunk/kernel/libc/strlen.c . . . 168
/home/phipse/gsoc/pok/trunk/kernel/middleware/portcreate.c . . . 168
/home/phipse/gsoc/pok/trunk/kernel/middleware/portflushall.c
Flush the ports and send the data of IN ports to OUT ports . . . 168
/home/phipse/gsoc/pok/trunk/kernel/middleware/portinit.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingcreate.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingid.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingreceive.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingsend.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingstatus.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingcreate.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingid.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingread.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingstatus.c . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingwrite.c
Send data on a sampling port . . . 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portutils.c
Various functions for ports management . . . 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualdestination.c . . . 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualgetglobal.c . . . 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualid.c . . . 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualnbdestinations.c . . . 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/queueinit.c . . . 170

/home/phipse/gsoc/pok/trunk/kernel/middleware/ressources.c 170

Chapter 3

Data Structure Documentation

3.1 __attribute__ Struct Reference

`#include <event.h>`

Data Fields

- uint32_t offset_low:16
- uint32_t segsel:16
- uint32_t res0:8
- uint32_t type:3
- uint32_t d:1
- uint32_t res1:1
- uint32_t dpl:2
- uint32_t present:1
- uint32_t offset_high:16
- uint32_t limit_low:16
- uint32_t base_low:24 __attribute__ ((packed))
- uint32_t s:1
- uint32_t limit_high:4
- uint32_t available:2
- uint32_t op_size:1
- uint32_t granularity:1
- uint32_t base_high:8
- uint16_t limit
- uint32_t base
- uint16_t padding
- uint32_t back_link
- uint32_t esp0
- uint32_t ss0
- uint32_t esp1
- uint32_t ss1
- uint32_t esp2
- uint32_t ss2
- uint32_t cr3
- uint32_t eip
- uint32_t eflags
- uint32_t eax
- uint32_t ecx

- uint32_t edx
- uint32_t ebx
- uint32_t esp
- uint32_t ebp
- uint32_t esi
- uint32_t edi
- uint32_t es
- uint32_t cs
- uint32_t ss
- uint32_t ds
- uint32_t fs
- uint32_t gs
- uint32_t ldt
- uint16_t trace_trap
- uint16_t io_bit_map_offset

3.1.1 Detailed Description

Definition at line 33 of file event.h.

3.1.2 Field Documentation

3.1.2.1 uint32_t __attribute__::available

Definition at line 39 of file gdt.h.

3.1.2.2 uint32_t __attribute__::back_link

Definition at line 25 of file tss.h.

3.1.2.3 uint32_t __attribute__::base

Definition at line 24 of file sysdesc.h.

3.1.2.4 uint32_t __attribute__::base_high

Definition at line 42 of file gdt.h.

3.1.2.5 uint32_t __attribute__::base_low

Definition at line 33 of file gdt.h.

3.1.2.6 uint32_t __attribute__::cr3

Definition at line 32 of file tss.h.

3.1.2.7 uint32_t __attribute__::cs

Definition at line 44 of file tss.h.

3.1.2.8 uint32_t __attribute__::d

Definition at line 39 of file event.h.

3.1.2.9 uint32_t __attribute__::dpl

Definition at line 41 of file event.h.

3.1.2.10 uint32_t __attribute__::ds

Definition at line 46 of file tss.h.

3.1.2.11 uint32_t __attribute__::eax

Definition at line 35 of file tss.h.

3.1.2.12 uint32_t __attribute__::ebp

Definition at line 40 of file tss.h.

3.1.2.13 uint32_t __attribute__::ebx

Definition at line 38 of file tss.h.

3.1.2.14 uint32_t __attribute__::ecx

Definition at line 36 of file tss.h.

3.1.2.15 uint32_t __attribute__::edi

Definition at line 42 of file tss.h.

3.1.2.16 uint32_t __attribute__::edx

Definition at line 37 of file tss.h.

3.1.2.17 uint32_t __attribute__::eflags

Definition at line 34 of file tss.h.

3.1.2.18 uint32_t __attribute__::eip

Definition at line 33 of file tss.h.

3.1.2.19 uint32_t __attribute__::es

Definition at line 43 of file tss.h.

3.1.2.20 uint32_t __attribute__::esi

Definition at line 41 of file tss.h.

3.1.2.21 uint32_t __attribute__::esp

Definition at line 39 of file tss.h.

3.1.2.22 uint32_t __attribute__::esp0

Definition at line 26 of file tss.h.

3.1.2.23 uint32_t __attribute__::esp1

Definition at line 28 of file tss.h.

3.1.2.24 uint32_t __attribute__::esp2

Definition at line 30 of file tss.h.

3.1.2.25 uint32_t __attribute__::fs

Definition at line 47 of file tss.h.

3.1.2.26 uint32_t __attribute__::granularity

Definition at line 41 of file gdt.h.

3.1.2.27 uint32_t __attribute__::gs

Definition at line 48 of file tss.h.

3.1.2.28 uint16_t __attribute__::io_bit_map_offset

Definition at line 51 of file tss.h.

3.1.2.29 uint32_t __attribute__::ldt

Definition at line 49 of file tss.h.

3.1.2.30 uint16_t __attribute__::limit

Definition at line 23 of file sysdesc.h.

3.1.2.31 uint32_t __attribute__::limit_high

Definition at line 38 of file gdt.h.

3.1.2.32 uint32_t __attribute__::limit_low

Definition at line 32 of file gdt.h.

3.1.2.33 uint32_t __attribute__::offset_high

Definition at line 43 of file event.h.

3.1.2.34 uint32_t __attribute__::offset_low

Definition at line 35 of file event.h.

3.1.2.35 uint32_t __attribute__::op_size

Definition at line 40 of file gdt.h.

3.1.2.36 uint16_t __attribute__::padding

Definition at line 25 of file sysdesc.h.

3.1.2.37 uint32_t __attribute__::present

Definition at line 42 of file event.h.

3.1.2.38 uint32_t __attribute__::res0

Definition at line 37 of file event.h.

3.1.2.39 uint32_t __attribute__::res1

Definition at line 40 of file event.h.

3.1.2.40 uint32_t __attribute__::s

Definition at line 35 of file gdt.h.

3.1.2.41 uint32_t __attribute__::segsel

Definition at line 36 of file event.h.

3.1.2.42 uint32_t __attribute__::ss

Definition at line 45 of file tss.h.

3.1.2.43 uint32_t __attribute__::ss0

Definition at line 27 of file tss.h.

3.1.2.44 uint32_t __attribute__::ss1

Definition at line 29 of file tss.h.

3.1.2.45 uint32_t __attribute__::ss2

Definition at line 31 of file tss.h.

3.1.2.46 uint16_t __attribute__::trace_trap

Definition at line 50 of file tss.h.

3.1.2.47 uint32_t __attribute__::type

Definition at line 38 of file event.h.

The documentation for this struct was generated from the following files:

- /home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.h
- /home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.h
- /home/phipse/gsoc/pok/trunk/kernel/arch/x86/sysdesc.h
- /home/phipse/gsoc/pok/trunk/kernel/arch/x86/tss.h

3.2 context_t Struct Reference

`#include <thread.h>`

Data Fields

- uint32_t sp
- uint32_t unused_lr
- uint32_t cr
- uint32_t r2
- uint32_t r13
- uint32_t r14
- uint32_t r15
- uint32_t r16
- uint32_t r17
- uint32_t r18
- uint32_t r19
- uint32_t r20
- uint32_t r21
- uint32_t r22
- uint32_t r23
- uint32_t r24
- uint32_t r25
- uint32_t r26
- uint32_t r27
- uint32_t r28
- uint32_t r29
- uint32_t r30
- uint32_t r31

- uint32_t pad
- uint32_t back_chain
- uint32_t lr
- uint32_t edi
- uint32_t esi
- uint32_t ebp
- uint32_t __esp
- uint32_t ebx
- uint32_t edx
- uint32_t ecx
- uint32_t eax
- uint32_t eip
- uint32_t cs
- uint32_t eflags

3.2.1 Detailed Description

Definition at line 23 of file thread.h.

3.2.2 Field Documentation

3.2.2.1 uint32_t context_t::__esp

Definition at line 28 of file thread.h.

3.2.2.2 uint32_t context_t::back_chain

Definition at line 54 of file thread.h.

3.2.2.3 uint32_t context_t::cr

Definition at line 28 of file thread.h.

3.2.2.4 uint32_t context_t::cs

Definition at line 35 of file thread.h.

3.2.2.5 uint32_t context_t::eax

Definition at line 32 of file thread.h.

3.2.2.6 uint32_t context_t::ebp

Definition at line 27 of file thread.h.

3.2.2.7 uint32_t context_t::ebx

Definition at line 29 of file thread.h.

3.2.2.8 uint32_t context_t::ecx

Definition at line 31 of file thread.h.

3.2.2.9 uint32_t context_t::edi

Definition at line 25 of file thread.h.

3.2.2.10 uint32_t context_t::edx

Definition at line 30 of file thread.h.

3.2.2.11 uint32_t context_t::eflags

Definition at line 36 of file thread.h.

3.2.2.12 uint32_t context_t::eip

Definition at line 34 of file thread.h.

3.2.2.13 uint32_t context_t::esi

Definition at line 26 of file thread.h.

3.2.2.14 uint32_t context_t::lr

Definition at line 55 of file thread.h.

3.2.2.15 uint32_t context_t::pad

Definition at line 51 of file thread.h.

3.2.2.16 uint32_t context_t::r13

Definition at line 30 of file thread.h.

3.2.2.17 uint32_t context_t::r14

Definition at line 31 of file thread.h.

3.2.2.18 uint32_t context_t::r15

Definition at line 32 of file thread.h.

3.2.2.19 uint32_t context_t::r16

Definition at line 34 of file thread.h.

3.2.2.20 uint32_t context_t::r17

Definition at line 35 of file thread.h.

3.2.2.21 uint32_t context_t::r18

Definition at line 36 of file thread.h.

3.2.2.22 uint32_t context_t::r19

Definition at line 37 of file thread.h.

3.2.2.23 uint32_t context_t::r2

Definition at line 29 of file thread.h.

3.2.2.24 uint32_t context_t::r20

Definition at line 38 of file thread.h.

3.2.2.25 uint32_t context_t::r21

Definition at line 39 of file thread.h.

3.2.2.26 uint32_t context_t::r22

Definition at line 40 of file thread.h.

3.2.2.27 uint32_t context_t::r23

Definition at line 41 of file thread.h.

3.2.2.28 uint32_t context_t::r24

Definition at line 42 of file thread.h.

3.2.2.29 uint32_t context_t::r25

Definition at line 43 of file thread.h.

3.2.2.30 uint32_t context_t::r26

Definition at line 44 of file thread.h.

3.2.2.31 uint32_t context_t::r27

Definition at line 45 of file thread.h.

3.2.2.32 uint32_t context_t::r28

Definition at line 46 of file thread.h.

3.2.2.33 uint32_t context_t::r29

Definition at line 47 of file thread.h.

3.2.2.34 uint32_t context_t::r30

Definition at line 48 of file thread.h.

3.2.2.35 uint32_t context_t::r31

Definition at line 49 of file thread.h.

3.2.2.36 uint32_t context_t::sp

Definition at line 25 of file thread.h.

3.2.2.37 uint32_t context_t::unused_lr

Definition at line 26 of file thread.h.

The documentation for this struct was generated from the following files:

- /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.h
- /home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.h

3.3 cpio_bin_header Struct Reference

`#include <cpio.h>`

Data Fields

- unsigned short c_magic
- unsigned short c_dev
- unsigned short c_ino
- unsigned short c_mode
- unsigned short c_uid
- unsigned short c_gid
- unsigned short c_nlink
- unsigned short c_rdev
- unsigned short c_mtime [2]
- unsigned short c_namesize
- unsigned short c_filesize [2]

3.3.1 Detailed Description

Definition at line 33 of file cpio.h.

3.3.2 Field Documentation

3.3.2.1 unsigned short cpio_bin_header::c_dev

Definition at line 36 of file cpio.h.

3.3.2.2 unsigned short cpio_bin_header::c_filesize[2]

Definition at line 45 of file cpio.h.

3.3.2.3 unsigned short cpio_bin_header::c_gid

Definition at line 40 of file cpio.h.

3.3.2.4 unsigned short cpio_bin_header::c_ino

Definition at line 37 of file cpio.h.

3.3.2.5 unsigned short cpio_bin_header::c_magic

Definition at line 35 of file cpio.h.

3.3.2.6 unsigned short cpio_bin_header::c_mode

Definition at line 38 of file cpio.h.

3.3.2.7 unsigned short cpio_bin_header::c_mtime[2]

Definition at line 43 of file cpio.h.

3.3.2.8 unsigned short cpio_bin_header::c_namesize

Definition at line 44 of file cpio.h.

3.3.2.9 unsigned short cpio_bin_header::c_nlink

Definition at line 41 of file cpio.h.

3.3.2.10 unsigned short cpio_bin_header::c_rdev

Definition at line 42 of file cpio.h.

3.3.2.11 unsigned short cpio_bin_header::c_uid

Definition at line 39 of file cpio.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/core/cpio.h

3.4 cpio_file Struct Reference

`#include <cpio.h>`

Data Fields

- int cpio_fmt
- void * cpio_addr
- void * curr_header
- void * curr_fileaddr
- unsigned int curr_filesz
- char * curr_filename
- unsigned int curr_filename_len
- int(* next_header)(struct cpio_file *cpio)

3.4.1 Detailed Description

Definition at line 48 of file cpio.h.

3.4.2 Field Documentation

3.4.2.1 void* cpio_file::cpio_addr

Definition at line 51 of file cpio.h.

3.4.2.2 int cpio_file::cpio_fmt

Definition at line 50 of file cpio.h.

3.4.2.3 void* cpio_file::curr_fileaddr

Definition at line 53 of file cpio.h.

3.4.2.4 char* cpio_file::curr_filename

Definition at line 55 of file cpio.h.

3.4.2.5 unsigned int cpio_file::curr_filename_len

Definition at line 56 of file cpio.h.

3.4.2.6 unsigned int cpio_file::curr_filesz

Definition at line 54 of file cpio.h.

3.4.2.7 void* cpio_file::curr_header

Definition at line 52 of file cpio.h.

3.4.2.8 int(* cpio_file::next_header)(struct cpio_file *cpio)

Definition at line 57 of file cpio.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/core/cpio.h

3.5 Elf32_Ehdr Struct Reference

`#include <elf.h>`

Data Fields

- unsigned char e_ident [EI_NIDENT]
- Elf32_Half e_type
- Elf32_Half e_machine
- Elf32_Word e_version
- Elf32_Addr e_entry
- Elf32_Off e_phoff
- Elf32_Off e_shoff
- Elf32_Word e_flags
- Elf32_Half e_ehsize
- Elf32_Half e_phentsize
- Elf32_Half e_phnum
- Elf32_Half e_shentsize
- Elf32_Half e_shnum
- Elf32_Half e_shstrndx

3.5.1 Detailed Description

Definition at line 28 of file elf.h.

3.5.2 Field Documentation

3.5.2.1 Elf32_Half Elf32_Ehdr::e_ehsize

Definition at line 38 of file elf.h.

3.5.2.2 Elf32_Addr Elf32_Ehdr::e_entry

Definition at line 34 of file elf.h.

3.5.2.3 Elf32_Word Elf32_Ehdr::e_flags

Definition at line 37 of file elf.h.

3.5.2.4 unsigned char Elf32_Ehdr::e_ident[EI_NIDENT]

Definition at line 30 of file elf.h.

3.5.2.5 Elf32_Half Elf32_Ehdr::e_machine

Definition at line 32 of file elf.h.

3.5.2.6 Elf32_Half Elf32_Ehdr::e_phentsize

Definition at line 39 of file elf.h.

3.5.2.7 Elf32_Half Elf32_Ehdr::e_phnum

Definition at line 40 of file elf.h.

3.5.2.8 Elf32_Off Elf32_Ehdr::e_phoff

Definition at line 35 of file elf.h.

3.5.2.9 Elf32_Half Elf32_Ehdr::e_shentsize

Definition at line 41 of file elf.h.

3.5.2.10 Elf32_Half Elf32_Ehdr::e_shnum

Definition at line 42 of file elf.h.

3.5.2.11 Elf32_Off Elf32_Ehdr::e_shoff

Definition at line 36 of file elf.h.

3.5.2.12 Elf32_Half Elf32_Ehdr::e_shstrndx

Definition at line 43 of file elf.h.

3.5.2.13 Elf32_Half Elf32_Ehdr::e_type

Definition at line 31 of file elf.h.

3.5.2.14 Elf32_Word Elf32_Ehdr::e_version

Definition at line 33 of file elf.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/elf.h

3.6 Elf32_Phdr Struct Reference

```
#include <elf.h>
```

Data Fields

- Elf32_Word p_type
- Elf32_Off p_offset
- Elf32_Addr p_vaddr
- Elf32_Addr p_paddr
- Elf32_Word p_filesz
- Elf32_Word p_memsz
- Elf32_Word p_flags
- Elf32_Word p_align

3.6.1 Detailed Description

Definition at line 48 of file elf.h.

3.6.2 Field Documentation

3.6.2.1 Elf32_Word Elf32_Phdr::p_align

Definition at line 57 of file elf.h.

3.6.2.2 Elf32_Word Elf32_Phdr::p_filesz

Definition at line 54 of file elf.h.

3.6.2.3 Elf32_Word Elf32_Phdr::p_flags

Definition at line 56 of file elf.h.

3.6.2.4 Elf32_Word Elf32_Phdr::p_memsz

Definition at line 55 of file elf.h.

3.6.2.5 Elf32_Off Elf32_Phdr::p_offset

Definition at line 51 of file elf.h.

3.6.2.6 Elf32_Addr Elf32_Phdr::p_paddr

Definition at line 53 of file elf.h.

3.6.2.7 Elf32_Word Elf32_Phdr::p_type

Definition at line 50 of file elf.h.

3.6.2.8 Elf32_Addr Elf32_Phdr::p_vaddr

Definition at line 52 of file elf.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/elf.h

3.7 interrupt_frame Struct Reference

`#include <interrupt.h>`

Data Fields

- uint32_t es
- uint32_t ds
- uint32_t edi
- uint32_t esi
- uint32_t ebp
- uint32_t __esp
- uint32_t ebx
- uint32_t edx
- uint32_t ecx
- uint32_t eax
- uint32_t error
- uint32_t eip
- uint32_t cs
- uint32_t eflags
- uint32_t esp
- uint32_t ss

3.7.1 Detailed Description

Definition at line 24 of file interrupt.h.

3.7.2 Field Documentation

3.7.2.1 uint32_t interrupt_frame::__esp

Definition at line 31 of file interrupt.h.

3.7.2.2 uint32_t interrupt_frame::cs

Definition at line 40 of file interrupt.h.

3.7.2.3 uint32_t interrupt_frame::ds

Definition at line 27 of file interrupt.h.

3.7.2.4 uint32_t interrupt_frame::eax

Definition at line 35 of file interrupt.h.

3.7.2.5 uint32_t interrupt_frame::ebp

Definition at line 30 of file interrupt.h.

3.7.2.6 uint32_t interrupt_frame::ebx

Definition at line 32 of file interrupt.h.

3.7.2.7 uint32_t interrupt_frame::ecx

Definition at line 34 of file interrupt.h.

3.7.2.8 uint32_t interrupt_frame::edi

Definition at line 28 of file interrupt.h.

3.7.2.9 uint32_t interrupt_frame::edx

Definition at line 33 of file interrupt.h.

3.7.2.10 uint32_t interrupt_frame::eflags

Definition at line 41 of file interrupt.h.

3.7.2.11 uint32_t interrupt_frame::eip

Definition at line 39 of file interrupt.h.

3.7.2.12 uint32_t interrupt_frame::error

Definition at line 38 of file interrupt.h.

3.7.2.13 uint32_t interrupt_frame::es

Definition at line 26 of file interrupt.h.

3.7.2.14 uint32_t interrupt_frame::esi

Definition at line 29 of file interrupt.h.

3.7.2.15 uint32_t interrupt_frame::esp

Definition at line 45 of file interrupt.h.

3.7.2.16 uint32_t interrupt_frame::ss

Definition at line 46 of file interrupt.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/interrupt.h

3.8 pok_aout_symbol_table_t Struct Reference

`#include <multiboot.h>`

Data Fields

- unsigned int tabsize
- unsigned int strsize
- unsigned int addr
- unsigned int reserved

3.8.1 Detailed Description

Definition at line 76 of file multiboot.h.

3.8.2 Field Documentation

3.8.2.1 unsigned int pok_aout_symbol_table_t::addr

Definition at line 80 of file multiboot.h.

3.8.2.2 unsigned int pok_aout_symbol_table_t::reserved

Definition at line 81 of file multiboot.h.

3.8.2.3 unsigned int pok_aout_symbol_table_t::strsize

Definition at line 79 of file multiboot.h.

3.8.2.4 unsigned int pok_aout_symbol_table_t::tabsize

Definition at line 78 of file multiboot.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h

3.9 pok_elf_section_header_table_t Struct Reference

`#include <multiboot.h>`

Data Fields

- unsigned int num
- unsigned int size
- unsigned int addr
- unsigned int shndx

3.9.1 Detailed Description

Definition at line 84 of file multiboot.h.

3.9.2 Field Documentation

3.9.2.1 unsigned int pok_elf_section_header_table_t::addr

Definition at line 88 of file multiboot.h.

3.9.2.2 unsigned int pok_elf_section_header_table_t::num

Definition at line 86 of file multiboot.h.

3.9.2.3 unsigned int pok_elf_section_header_table_t::shndx

Definition at line 89 of file multiboot.h.

3.9.2.4 unsigned int pok_elf_section_header_table_t::size

Definition at line 87 of file multiboot.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h

3.10 pok_lockobj_attr_t Struct Reference

`#include <lockobj.h>`

Data Fields

- pok_lockobj_kind_t kind
- pok_locking_policy_t locking_policy
- pok_queueing_discipline_t queueing_policy
- pok_sem_value_t initial_value
- pok_sem_value_t max_value

3.10.1 Detailed Description

Definition at line 63 of file lockobj.h.

3.10.2 Field Documentation

3.10.2.1 pok_sem_value_t pok_lockobj_attr_t::initial_value

Definition at line 68 of file lockobj.h.

3.10.2.2 pok_lockobj_kind_t pok_lockobj_attr_t::kind

Definition at line 65 of file lockobj.h.

3.10.2.3 pok_locking_policy_t pok_lockobj_attr_t::locking_policy

Definition at line 66 of file lockobj.h.

3.10.2.4 pok_sem_value_t pok_lockobj_attr_t::max_value

Definition at line 69 of file lockobj.h.

3.10.2.5 pok_queueing_discipline_t pok_lockobj_attr_t::queueing_policy

Definition at line 67 of file lockobj.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/core/lockobj.h

3.11 pok_lockobj_lockattr_t Struct Reference

`#include <lockobj.h>`

Data Fields

- pok_lockobj_operation_t operation
- pok_lockobj_kind_t obj_kind
- pok_lockobj_lock_kind_t lock_kind
- uint64_t time

3.11.1 Detailed Description

Definition at line 115 of file lockobj.h.

3.11.2 Field Documentation

3.11.2.1 pok_lockobj_lock_kind_t pok_lockobj_lockattr_t::lock_kind

Definition at line 119 of file lockobj.h.

3.11.2.2 pok_lockobj_kind_t pok_lockobj_lockattr_t::obj_kind

Definition at line 118 of file lockobj.h.

3.11.2.3 pok_lockobj_operation_t pok_lockobj_lockattr_t::operation

Definition at line 117 of file lockobj.h.

3.11.2.4 uint64_t pok_lockobj_lockattr_t::time

Definition at line 120 of file lockobj.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/core/lockobj.h

3.12 pok_lockobj_t Struct Reference

`#include <lockobj.h>`

Data Fields

- pok_spinlock_t spin
- pok_spinlock_t eventspin
- bool_t is_locked
- pok_mutex_state_t thread_state [POK_CONFIG_NB_THREADS+2]
- pok_locking_policy_t locking_policy
- pok_queueing_discipline_t queueing_policy
- pok_lockobj_kind_t kind
- bool_t initialized
- uint16_t current_value
- uint16_t max_value

3.12.1 Detailed Description

Definition at line 72 of file lockobj.h.

3.12.2 Field Documentation

3.12.2.1 uint16_t pok_lockobj_t::current_value

Definition at line 95 of file lockobj.h.

3.12.2.2 pok_spinlock_t pok_lockobj_t::eventspin

Definition at line 75 of file lockobj.h.

3.12.2.3 bool_t pok_lockobj_t::initialized

Definition at line 92 of file lockobj.h.

3.12.2.4 bool_t pok_lockobj_t::is_locked

Definition at line 78 of file lockobj.h.

3.12.2.5 pok_lockobj_kind_t pok_lockobj_t::kind

Definition at line 90 of file lockobj.h.

3.12.2.6 pok_locking_policy_t pok_lockobj_t::locking_policy

Definition at line 84 of file lockobj.h.

3.12.2.7 uint16_t pok_lockobj_t::max_value

Definition at line 96 of file lockobj.h.

3.12.2.8 pok_queueing_discipline_t pok_lockobj_t::queueing_policy

Definition at line 87 of file lockobj.h.

3.12.2.9 pok_spinlock_t pok_lockobj_t::spin

Definition at line 74 of file lockobj.h.

3.12.2.10 pok_mutex_state_t pok_lockobj_t::thread_state[POK_CONFIG_NB_THREADS+2]

Definition at line 81 of file lockobj.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/core/lockobj.h

3.13 pok_memory_map_t Struct Reference

`#include <multiboot.h>`

Data Fields

- unsigned int size
- unsigned int base_addr_low
- unsigned int base_addr_high
- unsigned int length_low
- unsigned int length_high
- unsigned int type

3.13.1 Detailed Description

Definition at line 120 of file multiboot.h.

3.13.2 Field Documentation

3.13.2.1 unsigned int pok_memory_map_t::base_addr_high

Definition at line 124 of file multiboot.h.

3.13.2.2 unsigned int pok_memory_map_t::base_addr_low

Definition at line 123 of file multiboot.h.

3.13.2.3 unsigned int pok_memory_map_t::length_high

Definition at line 126 of file multiboot.h.

3.13.2.4 unsigned int pok_memory_map_t::length_low

Definition at line 125 of file multiboot.h.

3.13.2.5 unsigned int pok_memory_map_t::size

Definition at line 122 of file multiboot.h.

3.13.2.6 unsigned int pok_memory_map_t::type

Definition at line 127 of file multiboot.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h

3.14 pok_module_t Struct Reference

`#include <multiboot.h>`

Data Fields

- unsigned int mod_start
- unsigned int mod_end
- unsigned int string
- unsigned int reserved

3.14.1 Detailed Description

Definition at line 112 of file multiboot.h.

3.14.2 Field Documentation

3.14.2.1 unsigned int pok_module_t::mod_end

Definition at line 115 of file multiboot.h.

3.14.2.2 unsigned int pok_module_t::mod_start

Definition at line 114 of file multiboot.h.

3.14.2.3 unsigned int pok_module_t::reserved

Definition at line 117 of file multiboot.h.

3.14.2.4 unsigned int pok_module_t::string

Definition at line 116 of file multiboot.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h

3.15 pok_multiboot_header_t Struct Reference

`#include <multiboot.h>`

Data Fields

- unsigned int magic
- unsigned int flags
- unsigned int checksum
- unsigned int header_addr
- unsigned int load_addr
- unsigned int load_end_addr
- unsigned int bss_end_addr
- unsigned int entry_addr

3.15.1 Detailed Description

Definition at line 64 of file multiboot.h.

3.15.2 Field Documentation

3.15.2.1 unsigned int pok_multiboot_header_t::bss_end_addr

Definition at line 72 of file multiboot.h.

3.15.2.2 unsigned int pok_multiboot_header_t::checksum

Definition at line 68 of file multiboot.h.

3.15.2.3 unsigned int pok_multiboot_header_t::entry_addr

Definition at line 73 of file multiboot.h.

3.15.2.4 unsigned int pok_multiboot_header_t::flags

Definition at line 67 of file multiboot.h.

3.15.2.5 unsigned int pok_multiboot_header_t::header_addr

Definition at line 69 of file multiboot.h.

3.15.2.6 unsigned int pok_multiboot_header_t::load_addr

Definition at line 70 of file multiboot.h.

3.15.2.7 unsigned int pok_multiboot_header_t::load_end_addr

Definition at line 71 of file multiboot.h.

3.15.2.8 unsigned int pok_multiboot_header_t::magic

Definition at line 66 of file multiboot.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h

3.16 pok_multiboot_info_t Struct Reference

`#include <multiboot.h>`

Data Fields

- unsigned int flags
- unsigned int mem_lower
- unsigned int mem_upper
- unsigned int boot_device
- unsigned int cmdline
- unsigned int mods_count
- unsigned int mods_addr
- union {
  pok_aout_symbol_table_t aout_sym
  pok_elf_section_header_table_t elf_sec
  } u
- unsigned int mmap_length
- unsigned int mmap_addr

3.16.1 Detailed Description

Definition at line 92 of file multiboot.h.

3.16.2 Field Documentation

3.16.2.1 pok_aout_symbol_table_t pok_multiboot_info_t::aout_sym

Definition at line 104 of file multiboot.h.

3.16.2.2 unsigned int pok_multiboot_info_t::boot_device

Definition at line 97 of file multiboot.h.

3.16.2.3 unsigned int pok_multiboot_info_t::cmdline

Definition at line 98 of file multiboot.h.

3.16.2.4 pok_elf_section_header_table_t pok_multiboot_info_t::elf_sec

Definition at line 105 of file multiboot.h.

3.16.2.5 unsigned int pok_multiboot_info_t::flags

Definition at line 94 of file multiboot.h.

3.16.2.6 unsigned int pok_multiboot_info_t::mem_lower

Definition at line 95 of file multiboot.h.

3.16.2.7 unsigned int pok_multiboot_info_t::mem_upper

Definition at line 96 of file multiboot.h.

3.16.2.8 unsigned int pok_multiboot_info_t::mmap_addr

Definition at line 109 of file multiboot.h.

3.16.2.9 unsigned int pok_multiboot_info_t::mmap_length

Definition at line 108 of file multiboot.h.

3.16.2.10 unsigned int pok_multiboot_info_t::mods_addr

Definition at line 100 of file multiboot.h.

3.16.2.11 unsigned int pok_multiboot_info_t::mods_count

Definition at line 99 of file multiboot.h.

3.16.2.12 union { ... } pok_multiboot_info_t::u

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h

3.17 pok_port_t Struct Reference

`#include <port.h>`

Data Fields

- pok_port_id_t identifier
- pok_partition_id_t partition
- pok_port_size_t index
- bool_t full
- pok_port_size_t size
- pok_port_size_t off_b
- pok_port_size_t off_e
- pok_port_direction_t direction
- pok_port_queueing_discipline_t discipline
- pok_bool_t ready
- bool_t empty
- uint8_t kind
- uint64_t refresh
- uint64_t last_receive
- pok_lockobj_t lock
- bool_t must_be_flushed

3.17.1 Detailed Description

Definition at line 57 of file port.h.

3.17.2 Field Documentation

3.17.2.1 pok_port_direction_t pok_port_t::direction

Definition at line 66 of file port.h.

3.17.2.2 pok_port_queueing_discipline_t pok_port_t::discipline

Definition at line 67 of file port.h.

3.17.2.3 bool_t pok_port_t::empty

Definition at line 69 of file port.h.

3.17.2.4 bool_t pok_port_t::full

Definition at line 62 of file port.h.

3.17.2.5 pok_port_id_t pok_port_t::identifier

Definition at line 59 of file port.h.

3.17.2.6 pok_port_size_t pok_port_t::index

Definition at line 61 of file port.h.

3.17.2.7 uint8_t pok_port_t::kind

Definition at line 70 of file port.h.

3.17.2.8 uint64_t pok_port_t::last_receive

Definition at line 72 of file port.h.

3.17.2.9 pok_lockobj_t pok_port_t::lock

Definition at line 73 of file port.h.

3.17.2.10 bool_t pok_port_t::must_be_flushed

Definition at line 74 of file port.h.

3.17.2.11 pok_port_size_t pok_port_t::off_b

Definition at line 64 of file port.h.

3.17.2.12 pok_port_size_t pok_port_t::off_e

Definition at line 65 of file port.h.

3.17.2.13 pok_partition_id_t pok_port_t::partition

Definition at line 60 of file port.h.

3.17.2.14 pok_bool_t pok_port_t::ready

Definition at line 68 of file port.h.

3.17.2.15 uint64_t pok_port_t::refresh

Definition at line 71 of file port.h.

3.17.2.16 pok_port_size_t pok_port_t::size

Definition at line 63 of file port.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/middleware/port.h

3.18 pok_space Struct Reference

Data Fields

- uint32_t phys_base
- uint32_t size

3.18.1 Detailed Description

Basic paritions informtation needed for memory managment.

Definition at line 34 of file space.c.

3.18.2 Field Documentation

3.18.2.1 uint32_t pok_space::phys_base

Pysical adress of the partition.

Definition at line 36 of file space.c.

3.18.2.2 uint32_t pok_space::size

Size of the partition.

Definition at line 37 of file space.c.

The documentation for this struct was generated from the following files:

- /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/space.c
- /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.c

3.19 pok_syscall_args_t Struct Reference

`#include <syscall.h>`

Data Fields

- uint32_t nargs
- uint32_t arg1
- uint32_t arg2
- uint32_t arg3
- uint32_t arg4
- uint32_t arg5

3.19.1 Detailed Description

Definition at line 92 of file syscall.h.

3.19.2 Field Documentation

3.19.2.1 uint32_t pok_syscall_args_t::arg1

Definition at line 95 of file syscall.h.

3.19.2.2 uint32_t pok_syscall_args_t::arg2

Definition at line 96 of file syscall.h.

3.19.2.3 uint32_t pok_syscall_args_t::arg3

Definition at line 97 of file syscall.h.

3.19.2.4 uint32_t pok_syscall_args_t::arg4

Definition at line 98 of file syscall.h.

3.19.2.5 uint32_t pok_syscall_args_t::arg5

Definition at line 99 of file syscall.h.

3.19.2.6 uint32_t pok_syscall_args_t::nargs

Definition at line 94 of file syscall.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/core/syscall.h

3.20 pok_syscall_info_t Struct Reference

`#include <syscall.h>`

Data Fields

- pok_partition_id_t partition
- uint32_t thread
- uint32_t base_addr

3.20.1 Detailed Description

Definition at line 102 of file syscall.h.

3.20.2 Field Documentation

3.20.2.1 uint32_t pok_syscall_info_t::base_addr

Definition at line 106 of file syscall.h.

3.20.2.2 pok_partition_id_t pok_syscall_info_t::partition

Definition at line 104 of file syscall.h.

3.20.2.3 uint32_t pok_syscall_info_t::thread

Definition at line 105 of file syscall.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/include/core/syscall.h

3.21 ppc_pte_t Struct Reference

Data Fields

- uint32_t vsid_api
- uint32_t rpn_flags

3.21.1 Detailed Description

Definition at line 109 of file space.c.

3.21.2 Field Documentation

3.21.2.1 uint32_t ppc_pte_t::rpn_flags

Definition at line 112 of file space.c.

3.21.2.2 uint32_t ppc_pte_t::vsid_api

Definition at line 111 of file space.c.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/space.c

3.22 space_context_t Struct Reference

```
#include <space.h>
```

Data Fields

- context_t ctx
- uint32_t fake_ret
- unsigned int partition_id
- uint32_t user_pc
- uint32_t user_sp
- uint32_t kernel_sp
- uint32_t arg1
- uint32_t arg2

3.22.1 Detailed Description

Definition at line 29 of file space.h.

3.22.2 Field Documentation

3.22.2.1 uint32_t space_context_t::arg1

Definition at line 37 of file space.h.

3.22.2.2 uint32_t space_context_t::arg2

Definition at line 38 of file space.h.

3.22.2.3 context_t space_context_t::ctx

Definition at line 31 of file space.h.

3.22.2.4 uint32_t space_context_t::fake_ret

Definition at line 32 of file space.h.

3.22.2.5 uint32_t space_context_t::kernel_sp

Definition at line 36 of file space.h.

3.22.2.6 unsigned int space_context_t::partition_id

Definition at line 33 of file space.h.

3.22.2.7 uint32_t space_context_t::user_pc

Definition at line 34 of file space.h.

3.22.2.8 uint32_t space_context_t::user_sp

Definition at line 35 of file space.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.h

3.23 start_context_t Struct Reference

`#include <thread.h>`

Data Fields

- context_t ctx
- uint32_t fake_ret
- uint32_t entry
- uint32_t id

3.23.1 Detailed Description

Definition at line 40 of file thread.h.

3.23.2 Field Documentation

3.23.2.1 context_t start_context_t::ctx

Definition at line 42 of file thread.h.

3.23.2.2 uint32_t start_context_t::entry

Definition at line 44 of file thread.h.

3.23.2.3 uint32_t start_context_t::fake_ret

Definition at line 43 of file thread.h.

3.23.2.4 uint32_t start_context_t::id

Definition at line 45 of file thread.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.h

3.24 volatile_context_t Struct Reference

`#include <thread.h>`

Data Fields

- uint32_t sp
- uint32_t unused_lr
- uint32_t cr
- uint32_t r0
- uint32_t r2
- uint32_t r3
- uint32_t r4
- uint32_t r5
- uint32_t r6
- uint32_t r7
- uint32_t r8
- uint32_t r9
- uint32_t r10
- uint32_t r11
- uint32_t r12
- uint32_t r13
- uint32_t ctr
- uint32_t xer
- uint32_t srr0
- uint32_t srr1
- uint32_t back_chain
- uint32_t lr
- uint32_t pad0
- uint32_t pad1

3.24.1 Detailed Description

Definition at line 58 of file thread.h.

3.24.2 Field Documentation

3.24.2.1 uint32_t volatile_context_t::back_chain

Definition at line 83 of file thread.h.

3.24.2.2 uint32_t volatile_context_t::cr

Definition at line 63 of file thread.h.

3.24.2.3 uint32_t volatile_context_t::ctr

Definition at line 77 of file thread.h.

3.24.2.4 uint32_t volatile_context_t::lr

Definition at line 84 of file thread.h.

3.24.2.5 uint32_t volatile_context_t::pad0

Definition at line 87 of file thread.h.

3.24.2.6 uint32_t volatile_context_t::pad1

Definition at line 88 of file thread.h.

3.24.2.7 uint32_t volatile_context_t::r0

Definition at line 64 of file thread.h.

3.24.2.8 uint32_t volatile_context_t::r10

Definition at line 73 of file thread.h.

3.24.2.9 uint32_t volatile_context_t::r11

Definition at line 74 of file thread.h.

3.24.2.10 uint32_t volatile_context_t::r12

Definition at line 75 of file thread.h.

3.24.2.11 uint32_t volatile_context_t::r13

Definition at line 76 of file thread.h.

3.24.2.12 uint32_t volatile_context_t::r2

Definition at line 65 of file thread.h.

3.24.2.13 uint32_t volatile_context_t::r3

Definition at line 66 of file thread.h.

3.24.2.14 uint32_t volatile_context_t::r4

Definition at line 67 of file thread.h.

3.24.2.15 uint32_t volatile_context_t::r5

Definition at line 68 of file thread.h.

3.24.2.16 uint32_t volatile_context_t::r6

Definition at line 69 of file thread.h.

3.24.2.17 uint32_t volatile_context_t::r7

Definition at line 70 of file thread.h.

3.24.2.18 uint32_t volatile_context_t::r8

Definition at line 71 of file thread.h.

3.24.2.19 uint32_t volatile_context_t::r9

Definition at line 72 of file thread.h.

3.24.2.20 uint32_t volatile_context_t::sp

Definition at line 60 of file thread.h.

3.24.2.21 uint32_t volatile_context_t::srr0

Definition at line 79 of file thread.h.

3.24.2.22 uint32_t volatile_context_t::srr1

Definition at line 80 of file thread.h.

3.24.2.23 uint32_t volatile_context_t::unused_lr

Definition at line 61 of file thread.h.

3.24.2.24 uint32_t volatile_context_t::xer

Definition at line 78 of file thread.h.

The documentation for this struct was generated from the following file:

- /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.h

Chapter 4

File Documentation

4.1 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/arch.c File Reference

Provide generic architecture access for PPC architecture.

```
#include <types.h>
#include <errno.h>
#include <core/partition.h>
#include "msr.h"
```

Functions

- void pok_arch_space_init (void)
- pok_ret_t pok_arch_init ()
- pok_ret_t pok_arch_preempt_disable ()
- pok_ret_t pok_arch_preempt_enable ()
- pok_ret_t pok_arch_idle ()
- pok_ret_t pok_arch_event_register (uint8_t vector, void(∗handler)(void))
- uint32_t pok_thread_stack_addr (const uint8_t partition_id, const uint32_t local_thread_id)

4.1.1 Detailed Description

Provide generic architecture access for PPC architecture.

Author

Tristan Gingold

Date

2009

Definition in file arch.c.

4.1.2 Function Documentation

4.1.2.1 pok_ret_t pok_arch_event_register (uint8_t *vector,* void(∗)(void) *handler*)

Register an event (for example, an interruption)

Definition at line 83 of file arch.c.

```
84 {
85   (void) vector;
86   (void) handler;
87
88   return (POK_ERRNO_OK);
89 }
```

4.1.2.2 pok_ret_t pok_arch_idle ( )

Function that do nothing. Useful for the idle task for example.

Definition at line 74 of file arch.c.

```
75 {
76    while (1)
77    {
78    }
79
80    return (POK_ERRNO_OK);
81 }
```

4.1.2.3 pok_ret_t pok_arch_init ( )

Function that initializes architecture concerns.

Definition at line 43 of file arch.c.

```
44 {
45   set_msr (MSR_IP);
46 #if POK_NEEDS_PARTITIONS
47   pok_arch_space_init();
48 #endif
49
50   return (POK_ERRNO_OK);
51 }
```

4.1.2.4 pok_ret_t pok_arch_preempt_disable ( )

Disable interruptions

Definition at line 53 of file arch.c.

```
54 {
55   unsigned int msr;
56
57   msr = get_msr();
58   msr &= ~MSR_EE;
59   set_msr(msr);
60   return (POK_ERRNO_OK);
61 }
```

4.1.2.5 pok_ret_t pok_arch_preempt_enable ( )

Enable interruptions

Definition at line 63 of file arch.c.

```
64 {
65   unsigned int msr;
66
67   msr = get_msr();
68   msr |= MSR_EE;
69   set_msr(msr);
70
71   return (POK_ERRNO_OK);
72 }
```

4.1.2.6 void pok_arch_space_init (void)

Initilize MMU tables.

Definition at line 132 of file space.c.

```
133 {
134   uint32_t sdr1;
135
136   pt_base = 0;
137   pt_mask = 0x3ff;
138
139   sdr1 = pt_base | (pt_mask >> 10);
140   asm volatile ("mtsdr1 %0" : : "r"(sdr1));
141 }
```

4.1.2.7 uint32_t pok_thread_stack_addr (const uint8_t *partition_id*, const uint32_t *local_thread_id*)

Returns the stack address for a the thread number N in a partition.

- partition_id indicates the partition that contains the thread.
- local_thread_id the thread-id of the thread inside the partition.

Returns

the stack address of the thread.

Definition at line 92 of file arch.c.

```
94 {
95    return pok_partitions[partition_id].size - 16 - (local_thread_id * POK_USER_STACK_SIZE);
96 }
```

4.2 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/arch.c File Reference

```
#include <types.h>
#include <errno.h>
#include <core/partition.h>
#include "traps.h"
#include "space.h"
#include "psr.h"
#include "sparc_conf.h"
#include "syscalls.h"
```

Functions

- pok_ret_t pok_arch_init ()
- pok_ret_t pok_arch_preempt_disable ()
- pok_ret_t pok_arch_preempt_enable ()
- pok_ret_t pok_arch_idle ()
- pok_ret_t pok_arch_event_register (uint8_t vector, void(∗handler)(void))
- uint32_t pok_thread_stack_addr (const uint8_t partition_id, const uint32_t local_thread_id)

4.2.1 Detailed Description

Author

Fabien Chouteau

Definition in file arch.c.

4.2.2 Function Documentation

4.2.2.1 pok_ret_t pok_arch_event_register (uint8_t *vector*, void(*)(void) *handler*)

Attach the handler to the given trap number (vector).

See Also

pok_sparc_isr

Definition at line 75 of file arch.c.

```
76 {
77   if (pok_sparc_isr[vector] == NULL)
78   {
79     pok_sparc_isr[vector] = handler;
80     return (POK_ERRNO_OK);
81   }
82   else
83   {
84     return (POK_ERRNO_UNAVAILABLE);
85   }
86 }
```

4.2.2.2 pok_ret_t pok_arch_idle ( )

Function that do nothing. Useful for the idle task for example.

Definition at line 60 of file arch.c.

```
61 {
62   while (1)
63   {
64     /* Leon3 Only ? */
65     asm volatile ("wr %g0, %asr19");
66   }
67
68    return (POK_ERRNO_OK);
69 }
```

4.2.2.3 pok_ret_t pok_arch_init ( )

Initialize all SPARC managers (traps, syscalls, space).

Definition at line 34 of file arch.c.

```
35 {
36   traps_init();
37   psr_disable_interupt();
38   psr_enable_traps();
39
40   pok_arch_space_init();
41   pok_syscalls_init();
42
43   return (POK_ERRNO_OK);
44 }
```

4.2.2.4 pok_ret_t pok_arch_preempt_disable ( )

Disable interruptions

Definition at line 46 of file arch.c.

```
47 {
48   psr_disable_interupt();
49
50   return (POK_ERRNO_OK);
51 }
```

4.2.2.5 pok_ret_t pok_arch_preempt_enable ( )

Enable interruptions

Definition at line 53 of file arch.c.

```
54 {
55   psr_enable_interupt();
56
57   return (POK_ERRNO_OK);
58 }
```

4.2.2.6 uint32_t pok_thread_stack_addr (const uint8_t *partition_id*, const uint32_t *local_thread_id*)

Compute the stack adress for the given thread.

Definition at line 91 of file arch.c.

```
93 {
94   return pok_partitions[partition_id].size - (local_thread_id * POK_USER_STACK_SIZE);
95 }
```

4.3 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/arch.c File Reference

Provides generic architecture interface for x86 architecture.

```
#include <errno.h>
#include <core/partition.h>
#include "event.h"
#include "gdt.h"
```

Functions

- pok_ret_t pok_arch_init ()
- pok_ret_t pok_arch_preempt_disable ()
- pok_ret_t pok_arch_preempt_enable ()
- pok_ret_t pok_arch_idle ()
- pok_ret_t pok_arch_event_register (uint8_t vector, void(∗handler)(void))
- uint32_t pok_thread_stack_addr (const uint8_t partition_id, const uint32_t local_thread_id)

4.3.1 Detailed Description

Provides generic architecture interface for x86 architecture.

Author

Julian Pidancet
Julien Delange

Definition in file arch.c.

4.3.2 Function Documentation

4.3.2.1 pok_ret_t pok_arch_event_register (uint8_t *vector*, void(*)(void) *handler*)

Register an event (for example, an interruption)

Attach the handler to the given trap number (vector).

See Also

pok_sparc_isr

Definition at line 60 of file arch.c.

```
62 {
63    pok_idt_set_gate (vector,
64                      GDT_CORE_CODE_SEGMENT << 3,
65                      (uint32_t)handler,
66                      IDTE_TRAP,
67                      3);
68
69    return (POK_ERRNO_OK);
70 }
```

4.3.2.2 pok_ret_t pok_arch_idle ( )

Function that do nothing. Useful for the idle task for example.

Definition at line 50 of file arch.c.

```
51 {
52    while (1)
53    {
54       asm ("hlt");
55    }
56
57    return (POK_ERRNO_OK);
58 }
```

4.3.2.3 pok_ret_t pok_arch_init ( )

Function that initializes architecture concerns.

Initialize all SPARC managers (traps, syscalls, space).

Definition at line 30 of file arch.c.

```
31 {
32   pok_gdt_init ();
33   pok_event_init ();
34
35   return (POK_ERRNO_OK);
36 }
```

4.3.2.4 pok_ret_t pok_arch_preempt_disable ( )

Disable interruptions

Definition at line 38 of file arch.c.

```
39 {
40   asm ("cli");
41   return (POK_ERRNO_OK);
42 }
```

4.3.2.5 pok_ret_t pok_arch_preempt_enable ( )

Enable interruptions

Definition at line 44 of file arch.c.

```
45 {
46   asm ("sti");
47   return (POK_ERRNO_OK);
48 }
```

4.3.2.6 uint32_t pok_thread_stack_addr (const uint8_t *partition_id*, const uint32_t *local_thread_id*)

Returns the stack address for a the thread number N in a partition.

- partition_id indicates the partition that contains the thread.
- local_thread_id the thread-id of the thread inside the partition.

Returns

the stack address of the thread.

Compute the stack adress for the given thread.

Definition at line 72 of file arch.c.

```
74 {
75    return pok_partitions[partition_id].size - 4 - (local_thread_id * POK_USER_STACK_SIZE);
76 }
```

4.4 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/msr.h File Reference

Macros

- #define MSR_DR (1 << 4)
- #define MSR_IR (1 << 5)
- #define MSR_IP (1 << 6)
- #define MSR_PR (1 << 14)
- #define MSR_EE (1 << 15)

4.4.1 Macro Definition Documentation

4.4.1.1 #define MSR_DR (1 << 4)

Definition at line 21 of file msr.h.

4.4.1.2 #define MSR_EE (1 << 15)

Definition at line 26 of file msr.h.

4.4.1.3 #define MSR_IP (1 << 6)

Definition at line 23 of file msr.h.

4.4.1.4 #define MSR_IR (1 << 5)

Definition at line 22 of file msr.h.

4.4.1.5 #define MSR_PR (1 << 14)

Definition at line 25 of file msr.h.

4.5 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/bsp.c File Reference

```
#include <errno.h>
#include <arch.h>
#include "cons.h"
```

Functions

- int pok_bsp_init (void)
- void * pok_bsp_mem_alloc (size_t sz)

Variables

- char _end []

4.5.1 Function Documentation

4.5.1.1 int pok_bsp_init (void)

Definition at line 22 of file bsp.c.

```
23 {
24    pok_cons_init ();
25
26    return (POK_ERRNO_OK);
27 }
```

4.5.1.2 void* pok_bsp_mem_alloc (size_t *sz*)

Definition at line 34 of file bsp.c.

```
35 {
36   char *res;
37
38   res = (char *)(((unsigned int)heap_end + 4095) & ~4095);
39   heap_end = res + sz;
40   return res;
41 }
```

4.5.2 Variable Documentation

4.5.2.1 char _end[]

4.6 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/bsp.c File Reference

```
#include <errno.h>
#include <arch.h>
#include <core/debug.h>
#include "cons.h"
#include "sparc_conf.h"
```

Functions

- int pok_bsp_init (void)
- void * pok_bsp_mem_alloc (size_t sz)

Variables

- char _end []

4.6.1 Detailed Description

Author

Fabien Chouteau

Definition in file bsp.c.

4.6.2 Function Documentation

4.6.2.1 int pok_bsp_init (void)

Definition at line 32 of file bsp.c.

```
33 {
34   pok_cons_init ();
35   return (POK_ERRNO_OK);
36 }
```

4.6.2.2 void* pok_bsp_mem_alloc (size_t *sz*)

Used for partition allocation. For SPARC support, all partitions are aligned on page size and all partition sizes have to be less than page size.

See Also

SPARC_PAGE_SIZE

Definition at line 44 of file bsp.c.

```
45 {
46   char *res;
47
48   /* Aligned on page size */
49   res = (char *)(((uint32_t)heap_end + SPARC_PAGE_SIZE) & ~ (
      SPARC_PAGE_SIZE - 1));
50   heap_end = res + sz;
51   return res;
52 }
```

4.6.3 Variable Documentation

4.6.3.1 char _end[ ]

4.7 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/bsp.c File Reference

```
#include <errno.h>
#include <arch.h>
#include "cons.h"
#include "pm.h"
#include "pit.h"
#include "pic.h"
```

Functions

- pok_ret_t pok_bsp_init (void)
- pok_ret_t pok_bsp_irq_acknowledge (uint8_t irq)
- pok_ret_t pok_bsp_irq_register (uint8_t irq, void(*handler)(void))
- void * pok_bsp_mem_alloc (size_t size)
- pok_ret_t pok_bsp_time_init ()

4.7.1 Function Documentation

4.7.1.1 pok_ret_t pok_bsp_init (void)

Definition at line 26 of file bsp.c.

```
27 {
28    pok_cons_init ();
29    pok_pm_init ();
30    pok_pic_init ();
31
32    return (POK_ERRNO_OK);
33 }
```

4.7.1.2 pok_ret_t pok_bsp_irq_acknowledge (uint8_t *irq*)

Definition at line 35 of file bsp.c.

```
36 {
37    pok_pic_eoi (irq);
38
39    return (POK_ERRNO_OK);
40 }
```

4.7.1.3 pok_ret_t pok_bsp_irq_register (uint8_t *irq*, void(∗)(void) *handler*)

Definition at line 42 of file bsp.c.

```
44 {
45    pok_pic_unmask (irq);
46
47    pok_arch_event_register (32 + irq, handler);
48
49    return (POK_ERRNO_OK);
50 }
```

4.7.1.4 void∗ pok_bsp_mem_alloc (size_t *size*)

Allocate data. At this time, the pok_pm_sbrk function only increment size each time we allocate memory and was not designed to free previously allocated memory.

Definition at line 58 of file bsp.c.

```
59 {
60    return ((void *)pok_pm_sbrk(size));
61 }
```

4.7.1.5 pok_ret_t pok_bsp_time_init ( )

Init time. *freq* is the frequency of the oscillator.

Definition at line 67 of file bsp.c.

```
68 {
69    return (pok_x86_qemu_timer_init ());
70 }
```

4.8 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/cons.c File Reference

```
#include <errno.h>
#include "ioports.h"
#include <libc.h>
#include <core/debug.h>
#include <core/cons.h>
#include "cons.h"
```

Functions

- int pok_cons_init (void)

4.8.1 Function Documentation

4.8.1.1 int pok_cons_init (void)

Definition at line 68 of file cons.c.

```
69 {
70    return 0;
71 }
```

4.9 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/cons.c File Reference

Leon3 UART driver.

```
#include <errno.h>
#include "ioports.h"
#include <libc.h>
#include <core/debug.h>
#include <core/cons.h>
#include "cons.h"
```

Functions

- int pok_cons_init (void)

4.9.1 Detailed Description

Leon3 UART driver.

Author

Fabien Chouteau

Definition in file cons.c.

4.9.2 Function Documentation

4.9.2.1 int pok_cons_init (void)

Definition at line 62 of file cons.c.

```
63 {
64   return 0;
65 }
```

4.10 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/cons.c File Reference

```
#include <errno.h>
#include <arch/x86/ioports.h>
#include <libc.h>
#include <core/debug.h>
#include <core/cons.h>
#include "cons.h"
```

Functions

- int pok_cons_init (void)

4.10.1 Function Documentation

4.10.1.1 int pok_cons_init (void)

Definition at line 235 of file cons.c.

```
236 {
237     return 0;
238 }
```

4.11 /home/phipse/gsoc/pok/trunk/kernel/core/cons.c File Reference

4.12 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/cons.h File Reference

Functions

- int pok_cons_init (void)

4.12.1 Function Documentation

4.12.1.1 int pok_cons_init (void)

Definition at line 68 of file cons.c.

```
69 {
70     return 0;
71 }
```

4.13 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/cons.h File Reference

Macros

- #define UART_STATUS_DR 0x00000001
- #define UART_STATUS_TSE 0x00000002
- #define UART_STATUS_THE 0x00000004
- #define UART_STATUS_BR 0x00000008
- #define UART_STATUS_OE 0x00000010
- #define UART_STATUS_PE 0x00000020
- #define UART_STATUS_FE 0x00000040
- #define UART_STATUS_ERR 0x00000078
- #define UART_CTRL_RE 0x00000001
- #define UART_CTRL_TE 0x00000002
- #define UART_CTRL_RI 0x00000004
- #define UART_CTRL_TI 0x00000008
- #define UART_CTRL_PS 0x00000010
- #define UART_CTRL_PE 0x00000020
- #define UART_CTRL_FL 0x00000040
- #define UART_CTRL_LB 0x00000080
- #define UART_DATA_OFFSET 0x0
- #define UART_STAT_OFFSET 0x4
- #define UART_CTRL_OFFSET 0x8
- #define UART_SCALER_OFFSET 0xc
- #define UART1 0x80000100

Functions

- int pok_cons_init (void)

4.13.1 Detailed Description

Author

Fabien Chouteau

Definition in file cons.h.

4.13.2 Macro Definition Documentation

4.13.2.1 #define UART1 0x80000100

First Leon3 UART IO adress

Definition at line 48 of file cons.h.

4.13.2.2 #define UART_CTRL_FL 0x00000040

Flow control enable

Definition at line 40 of file cons.h.

4.13.2.3 #define UART_CTRL_LB 0x00000080

Loop Back enable

Definition at line 41 of file cons.h.

4.13.2.4 #define UART_CTRL_OFFSET 0x8

Control register offset

Definition at line 45 of file cons.h.

4.13.2.5 #define UART_CTRL_PE 0x00000020

Parity enable

Definition at line 39 of file cons.h.

4.13.2.6 #define UART_CTRL_PS 0x00000010

Parity select

Definition at line 38 of file cons.h.

4.13.2.7 #define UART_CTRL_RE 0x00000001

Receiver enable

Definition at line 34 of file cons.h.

4.13.2.8 #define UART_CTRL_RI 0x00000004

Receiver interrupt enable

Definition at line 36 of file cons.h.

4.13.2.9 #define UART_CTRL_TE 0x00000002

Transmitter enable

Definition at line 35 of file cons.h.

4.13.2.10 #define UART_CTRL_TI 0x00000008

Transmitter interrupt enable

Definition at line 37 of file cons.h.

4.13.2.11 #define UART_DATA_OFFSET 0x0

Data register offset

Definition at line 43 of file cons.h.

4.13.2.12 #define UART_SCALER_OFFSET 0xc

Scaler register offset

Definition at line 46 of file cons.h.

4.13.2.13 #define UART_STAT_OFFSET 0x4

Stat register offset

Definition at line 44 of file cons.h.

4.13.2.14 #define UART_STATUS_BR 0x00000008

Break Error

Definition at line 28 of file cons.h.

4.13.2.15 #define UART_STATUS_DR 0x00000001

Data Ready

Definition at line 25 of file cons.h.

4.13.2.16 #define UART_STATUS_ERR 0x00000078

Error Mask

Definition at line 32 of file cons.h.

4.13.2.17 #define UART_STATUS_FE 0x00000040

RX Framing Error

Definition at line 31 of file cons.h.

4.13.2.18 #define UART_STATUS_OE 0x00000010

RX Overrun Error

Definition at line 29 of file cons.h.

4.13.2.19 #define UART_STATUS_PE 0x00000020

RX Parity Error

Definition at line 30 of file cons.h.

4.13.2.20 #define UART_STATUS_THE 0x00000004

TX Hold Register Empty

Definition at line 27 of file cons.h.

4.13.2.21 #define UART_STATUS_TSE 0x00000002

TX Send Register Empty

Definition at line 26 of file cons.h.

4.13.3 Function Documentation

4.13.3.1 int pok_cons_init (void)

Definition at line 68 of file cons.c.

```
69 {
70     return 0;
71 }
```

4.14 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/cons.h File Reference

Functions

- int pok_cons_init (void)

4.14.1 Function Documentation

4.14.1.1 int pok_cons_init (void)

Definition at line 68 of file cons.c.

```
69 {
70     return 0;
71 }
```

4.15 /home/phipse/gsoc/pok/trunk/kernel/include/core/cons.h File Reference

4.16 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/ioports.h File Reference

Macros

- #define POK_PREP_IOBASE 0x80000000
- #define outb(port, data) *((volatile unsigned char *)(POK_PREP_IOBASE + port)) = data
- #define inb(port) *((volatile unsigned char *)(POK_PREP_IOBASE + port))

4.16.1 Macro Definition Documentation

4.16.1.1 #define inb(*port*) *((volatile unsigned char *)(POK_PREP_IOBASE + port))

Definition at line 26 of file ioports.h.

4.16.1.2 #define outb(*port, data*) *((volatile unsigned char *)(POK_PREP_IOBASE + port)) = data

Definition at line 23 of file ioports.h.

4.16.1.3 #define POK_PREP_IOBASE 0x80000000

Definition at line 21 of file ioports.h.

4.17 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/ioports.h File Reference

SPARC "ioports". Use MMU bypass to access IO memory.

```
#include <types.h>
#include "sparc_conf.h"
```

4.17.1 Detailed Description

SPARC "ioports". Use MMU bypass to access IO memory.

Author

Fabien Chouteau

Definition in file ioports.h.

4.18 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/ioports.h File Reference

```
#include <core/syscall.h>
```

Macros

- #define outb(port, data)
- #define inb(port)
- #define outl(port, data)
- #define inl(port)

4.18.1 Macro Definition Documentation

4.18.1.1 #define inb(*port*)

Value:

```
({                                                      \
  unsigned char res;                                    \
  asm volatile ("inb %w1,%0"                            \
                :"=a" (res)                             \
                :"d" (port));                           \
  res;                                                  \
})
```

Definition at line 28 of file ioports.h.

4.18.1.2 #define inl(*port*)

Value:

```
({                                                      \
  unsigned int res;                                     \
  asm volatile ("inl %w1,%0"                            \
                :"=a" (res)                             \
                :"d" (port));                           \
  res;                                                  \
})
```

Definition at line 42 of file ioports.h.

4.18.1.3 #define outb(*port, data*)

Value:

```
asm volatile ("outb %b0,%w1"                            \
                :                                       \
                :"a" (data),"d" (port))
```

Definition at line 23 of file ioports.h.

4.18.1.4 #define outl(*port, data*)

Value:

```
asm volatile ("outl %0,%w1"                             \
                :                                       \
                :"a" (data),"d" (port))
```

Definition at line 37 of file ioports.h.

4.19 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/space.c File Reference

```
#include <types.h>
#include <errno.h>
#include <libc.h>
#include <bsp.h>
#include <core/sched.h>
#include <arch.h>
#include "thread.h"
#include "msr.h"
```

Data Structures

- struct pok_space
- struct ppc_pte_t

Macros

- #define KERNEL_STACK_SIZE 8192
- #define PPC_SR_KP (1 << 29)
- #define PPC_SR_Ks (1 << 30)
- #define PPC_SR_T (1 << 31)
- #define PPC_PTE_V (1 << 31)
- #define POK_PAGE_SIZE (1 << 12)
- #define POK_PAGE_MASK (~(POK_PAGE_SIZE - 1))
- #define PPC_PTE_H (1 << 6)
- #define PPC_PTE_R (1 << 8)
- #define PPC_PTE_C (1 << 7)
- #define PPC_PTE_W (1 << 6)
- #define PPC_PTE_I (1 << 5)
- #define PPC_PTE_M (1 << 4)
- #define PPC_PTE_G (1 << 3)
- #define PPC_PTE_PP_NO 0
- #define PPC_PTE_PP_RO 1
- #define PPC_PTE_PP_RW 2

Functions

- pok_ret_t pok_create_space (uint8_t partition_id, uint32_t addr, uint32_t size)
- pok_ret_t pok_space_switch (uint8_t old_partition_id, uint8_t new_partition_id)
- uint32_t pok_space_base_vaddr (uint32_t addr)
- void pok_arch_rfi (void)
- uint32_t pok_space_context_create (uint8_t partition_id, uint32_t entry_rel, uint32_t stack_rel, uint32_t arg1, uint32_t arg2)
- void pok_arch_space_init (void)
- void pok_arch_isi_int (uint32_t pc, uint32_t msr)
- void pok_arch_dsi_int (uint32_t dar, uint32_t dsisr)

Variables

- struct pok_space spaces [POK_CONFIG_NB_PARTITIONS]

4.19.1 Macro Definition Documentation

4.19.1.1 #define KERNEL_STACK_SIZE 8192

Definition at line 28 of file space.c.

4.19.1.2 #define POK_PAGE_MASK (~(POK_PAGE_SIZE - 1))

Definition at line 120 of file space.c.

4.19.1.3 #define POK_PAGE_SIZE (1 << 12)

Definition at line 119 of file space.c.

4.19.1.4 #define PPC_PTE_C (1 << 7)

Definition at line 123 of file space.c.

4.19.1.5 #define PPC_PTE_G (1 << 3)

Definition at line 127 of file space.c.

4.19.1.6 #define PPC_PTE_H (1 << 6)

Definition at line 121 of file space.c.

4.19.1.7 #define PPC_PTE_I (1 << 5)

Definition at line 125 of file space.c.

4.19.1.8 #define PPC_PTE_M (1 << 4)

Definition at line 126 of file space.c.

4.19.1.9 #define PPC_PTE_PP_NO 0

Definition at line 128 of file space.c.

4.19.1.10 #define PPC_PTE_PP_RO 1

Definition at line 129 of file space.c.

4.19.1.11 #define PPC_PTE_PP_RW 2

Definition at line 130 of file space.c.

4.19.1.12 #define PPC_PTE_R (1 << 8)

Definition at line 122 of file space.c.

4.19.1.13 #define PPC_PTE_V (1 << 31)

Definition at line 118 of file space.c.

4.19.1.14 #define PPC_PTE_W (1 << 6)

Definition at line 124 of file space.c.

4.19.1.15 #define PPC_SR_KP (1 << 29)

Definition at line 30 of file space.c.

4.19.1.16 #define PPC_SR_Ks (1 << 30)

Definition at line 31 of file space.c.

4.19.1.17 #define PPC_SR_T (1 << 31)

Definition at line 32 of file space.c.

4.19.2 Function Documentation

4.19.2.1 void pok_arch_dsi_int (uint32_t *dar*, uint32_t *dsisr*)

Definition at line 203 of file space.c.

```
204 {
205 #ifdef POK_NEEDS_DEBUG
206   printf("dsi_int: part=%d, dar=%x dsisr=%x\n",
207          pok_current_partition, dar, dsisr);
208 #endif
209
210   if (dsisr & (1 << 30))
211     {
212       /* Page fault  */
213       if (dar < spaces[pok_current_partition].size)
214         {
215           uint32_t vaddr = dar & POK_PAGE_MASK;
216           uint32_t v;
217           v = (spaces[pok_current_partition].phys_base + vaddr) & POK_PAGE_MASK;
218           v |= PPC_PTE_R | PPC_PTE_C | PPC_PTE_PP_RW;
219           pok_insert_pte (pok_current_partition, vaddr, v);
220           return;
221         }
222     }
223 #ifdef POK_NEEDS_DEBUG
224    printf("[DEBUG] Infinite loop in pok_arch_dsi_int\n");
225 #endif
226   while (1)
227     ;
228 }
```

4.19.2.2 void pok_arch_isi_int (uint32_t *pc*, uint32_t *msr*)

Definition at line 168 of file space.c.

```
169 {
170
171 #ifdef POK_NEEDS_DEBUG
172   printf("isi_int: part=%d, pc=%x msr=%x\n",
173          pok_current_partition, pc, msr);
174
175   if (msr & ((1 << 28) | (1 << 27)))
```

```
176   {
177     printf (" Bad access\n");
178   }
179 #endif
180
181   if (msr & (1 << 30))
182     {
183       /* Page fault  */
184       if (pc < spaces[pok_current_partition].size)
185         {
186           uint32_t vaddr = pc & POK_PAGE_MASK;
187           uint32_t v;
188           v = (spaces[pok_current_partition].phys_base + vaddr) & POK_PAGE_MASK;
189           v |= PPC_PTE_R | PPC_PTE_C | PPC_PTE_PP_RW;
190           pok_insert_pte (pok_current_partition, vaddr, v);
191           return;
192         }
193     }
194
195 #ifdef POK_NEEDS_DEBUG
196    printf("[DEBUG] Infinite loop in pok_arch_isi_int\n");
197 #endif
198
199   while (1)
200     ;
201 }
```

4.19.2.3 void pok_arch_rfi (void)

4.19.2.4 void pok_arch_space_init (void)

Definition at line 132 of file space.c.

```
133 {
134   uint32_t sdr1;
135
136   pt_base = 0;
137   pt_mask = 0x3ff;
138
139   sdr1 = pt_base | (pt_mask >> 10);
140   asm volatile ("mtsdr1 %0" : : "r"(sdr1));
141 }
```

4.19.2.5 pok_ret_t pok_create_space (uint8_t *partition_id*, uint32_t *addr*, uint32_t *size*)

Definition at line 42 of file space.c.

```
45 {
46 #ifdef POK_NEEDS_DEBUG
47   printf ("pok_create_space: %d: %x %x\n", partition_id, addr, size);
48 #endif
49   spaces[partition_id].phys_base = addr;
50   spaces[partition_id].size = size;
51
52   return (POK_ERRNO_OK);
53 }
```

4.19.2.6 uint32_t pok_space_base_vaddr (uint32_t *addr*)

Definition at line 64 of file space.c.

```
65 {
66    (void) addr;
67    return (0);
68 }
```

4.19.2.7 uint32_t pok_space_context_create (uint8_t *partition_id*, uint32_t *entry_rel*, uint32_t *stack_rel*, uint32_t *arg1*, uint32_t *arg2*)

Create a new context in the given space

Definition at line 72 of file space.c.

```
77 {
78   context_t* ctx;
79   volatile_context_t* vctx;
80   char*      stack_addr;
81   (void) partition_id;
82
83   stack_addr = pok_bsp_mem_alloc (KERNEL_STACK_SIZE);
84
85   vctx = (volatile_context_t *)
86     (stack_addr + KERNEL_STACK_SIZE - sizeof (
      volatile_context_t));
87   ctx = (context_t *)((char *)vctx - sizeof (context_t) + 8);
88
89   memset (ctx, 0, sizeof (*ctx));
90   memset (vctx, 0, sizeof (*vctx));
91
92   vctx->r3     = arg1;
93   vctx->r4     = arg2;
94   vctx->sp     = stack_rel - 12;
95   vctx->srr0   = entry_rel;
96   vctx->srr1   = MSR_EE | MSR_IP | MSR_DR | MSR_IR |
      MSR_PR;
97   ctx->lr      = (uint32_t) pok_arch_rfi;
98
99   ctx->sp      = (uint32_t) &vctx->sp;
100
101 #ifdef POK_NEEDS_DEBUG
102   printf ("space_context_create %d: entry=%x stack=%x arg1=%x arg2=%x ksp=%x\n",
103           partition_id, entry_rel, stack_rel, arg1, arg2, &vctx->sp);
104 #endif
105
106   return (uint32_t)ctx;
107 }
```

4.19.2.8 pok_ret_t pok_space_switch (uint8_t *old_partition_id*, uint8_t *new_partition_id*)

Switch from one space to another

Definition at line 55 of file space.c.

```
57 {
58   (void) old_partition_id;
59   /* printf ("space_switch %u -> %u\n", old_partition_id, new_partition_id); */
60   asm volatile ("mtsr %0,%1" : : "r"(0), "r"(PPC_SR_KP | new_partition_id));
61   return (POK_ERRNO_OK);
62 }
```

4.19.3 Variable Documentation

4.19.3.1 struct pok_space spaces[POK_CONFIG_NB_PARTITIONS]

Definition at line 40 of file space.c.

4.20 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.c File Reference

Memory management in SPARC.

```
#include <types.h>
#include <errno.h>
#include <libc.h>
#include <bsp.h>
#include <core/sched.h>
#include <arch.h>
#include "thread.h"
#include "space.h"
#include "sparc_conf.h"
#include "context_offset.h"
#include "ioports.h"
```

Data Structures

- struct pok_space

Macros

- #define KERNEL_STACK_SIZE 8192

Functions

- ptd mmu_contexts_tab[POK_CONFIG_NB_PARTITIONS] __attribute__ ((aligned(POK_CONFIG_NB_PARTITIONS ∗sizeof(ptd))))
- ptd mmu_level1_tab[POK_CONFIG_NB_PARTITIONS][MM_LVL1_ENTRIES_NBR] __attribute__ ((aligned(MM_LVL1_ENTRIES_NBR ∗sizeof(ptd))))
- pte mmu_level2_tab[POK_CONFIG_NB_PARTITIONS][MM_LVL2_ENTRIES_NBR] __attribute__ ((aligned(MM_LVL2_ENTRIES_NBR ∗sizeof(pte))))
- pok_ret_t pok_create_space (uint8_t partition_id, uint32_t addr, uint32_t size)
- pok_ret_t pok_space_switch (uint8_t old_partition_id, uint8_t new_partition_id)
- uint32_t pok_space_base_vaddr (uint32_t addr)
- uint32_t pok_space_context_create (uint8_t id, uint32_t entry_rel, uint32_t stack_rel, uint32_t arg1, uint32_t arg2)
- void pok_arch_space_init (void)

Variables

- struct pok_space spaces [POK_CONFIG_NB_PARTITIONS]

4.20.1 Detailed Description

Memory management in SPARC.

Author

Fabien Chouteau

Definition in file space.c.

4.20.2 Macro Definition Documentation

4.20.2.1 #define KERNEL_STACK_SIZE 8192

Definition at line 36 of file space.c.

4.20.3 Function Documentation

4.20.3.1 ptd mmu_contexts_tab [POK_CONFIG_NB_PARTITIONS] __attribute__ ((aligned(POK_CONFIG_NB_PARTITIONS ∗sizeof(ptd))))

MMU contexts table. (cf SPARC V8 Manual, page 243)

4.20.3.2 ptd mmu_level1_tab [POK_CONFIG_NB_PARTITIONS][MM_LVL1_ENTRIES_NBR] __attribute__ ((aligned(MM_LVL1_ENTRIES_NBR ∗sizeof(ptd))))

MMU level 1 table. (cf SPARC V8 Manual, page 243)

4.20.3.3 pte mmu_level2_tab [POK_CONFIG_NB_PARTITIONS][MM_LVL2_ENTRIES_NBR] __attribute__ ((aligned(MM_LVL2_ENTRIES_NBR ∗sizeof(pte))))

MMU level 2 table. (cf SPARC V8 Manual, page 243)

4.20.3.4 void pok_arch_space_init (void)

Initilize MMU tables.

Definition at line 159 of file space.c.

```
160 {
161   int i = 0;
162   int j = 0;
163
164   for (i = 0; i < POK_CONFIG_NB_PARTITIONS; i++)
165     mmu_contexts_tab[i] = MM_ET_INVALID;
166
167   for (i = 0; i < POK_CONFIG_NB_PARTITIONS; i++)
168   {
169     mmu_contexts_tab[i] = (unsigned int)&(mmu_level1_tab[i]) >> 4 | MM_ET_PTD;
170
171     for (j = 0; j < MM_LVL1_ENTRIES_NBR; j++)
172     {
173       mmu_level1_tab[i][j] = MM_ET_INVALID;
174     }
175
176     for (j = 0; j < MM_LVL2_ENTRIES_NBR; j++)
177     {
178       mmu_level2_tab[i][j] = MM_ET_INVALID;
179     }
180   }
181
182   unsigned int kernel_pte = mm_index1(SPARC_RAM_ADDR);
183
184   /* the kernel code is always mapped on a 16Mb page (including all partitions) */
185   for (i = 0; i < POK_CONFIG_NB_PARTITIONS; i++)
186   {
187     mmu_level1_tab[i][kernel_pte] = (SPARC_RAM_ADDR >> 4) |
      MM_ACC_S_RWE | MM_ET_PTE | MM_CACHEABLE;
188   }
189
190
191   /* set context table */
192   asm volatile ("sta %0, [%1] %2;\n"
193                 : /* no output */
194                 : "r" (((unsigned int) mmu_contexts_tab) >> 4), "r" (
      MMU_CTXTBL_PTR), "i" (ASI_M_MMUREGS)
195                 : "memory");
196
197   /* set context number */
198   pok_space_switch(0, 0);
199
200   asm volatile ("flush\n"
201                 "sta %0, [%1] %2;\n"
202                 : /* no output */
203                 : "r" (0x1), "r" (MMU_CTRL_REG), "i" (ASI_M_MMUREGS)
204                 : "memory");
205
206
207 #ifdef POK_NEEDS_DEBUG
```

```
208     printf ("pok_arch_space_init: ctx nbr=%u\n", POK_CONFIG_NB_PARTITIONS);
209 #endif
210 }
```

4.20.3.5 pok_ret_t pok_create_space (uint8_t *partition_id*, uint32_t *addr*, uint32_t *size*)

Set ptd and pte for the given partition.

Definition at line 70 of file space.c.

```
73 {
74   if (size > SPARC_PARTITION_SIZE)
75   {
76 #ifdef POK_NEEDS_DEBUG
77     printf ("pok_create_space: %d: partition size too big 0x%x\n", partition_id, size);
78 #endif
79     return (POK_ERRNO_SIZE);
80   }
81
82   if ((addr & (SPARC_PAGE_SIZE - 1)) != 0)
83   {
84 #ifdef POK_NEEDS_DEBUG
85     printf ("pok_create_space: %d: partition address not aligned 0x%x\n", partition_id, addr);
86 #endif
87     return (POK_ERRNO_EFAULT);
88   }
89 #ifdef POK_NEEDS_DEBUG
90   printf ("pok_create_space: %d: %x %x\n", partition_id, addr, size);
91 #endif
92   spaces[partition_id].phys_base = addr;
93   spaces[partition_id].size = size;
94
95   unsigned int as_ptd = mm_index1(SPARC_PARTITION_BASE_VADDR);
96   unsigned int as_pte = mm_index2(SPARC_PARTITION_BASE_VADDR);
97
98   mmu_level1_tab[partition_id][as_ptd] = ((unsigned int) &(mmu_level2_tab[partition_id]) >> 4) |
      MM_ET_PTD;
99   /* partition as */
100    mmu_level2_tab[partition_id][as_pte] = ((addr) >> 4) | MM_ACC_RWE |
      MM_ET_PTE | MM_CACHEABLE;
101
102   return (POK_ERRNO_OK);
103 }
```

4.20.3.6 uint32_t pok_space_base_vaddr (uint32_t *addr*)

Returns

partition virtual base adress.

See Also

SPARC_PARTITION_BASE_VADDR

Definition at line 125 of file space.c.

```
126 {
127    (void) addr;
128    return (SPARC_PARTITION_BASE_VADDR);
129 }
```

4.20.3.7 uint32_t pok_space_context_create (uint8_t *id*, uint32_t *entry_rel*, uint32_t *stack_rel*, uint32_t *arg1*, uint32_t *arg2*)

Initilize thread stack.

Definition at line 134 of file space.c.

```
139 {
140   uint32_t ctx = spaces[id].phys_base + stack_rel - 0x40;
141
142   outw(ctx - RESTORE_CNT_OFFSET, 1); /* Only 1 register window needed */
143   outw(ctx - PC_OFFSET, entry_rel);
144   outw(ctx - NPC_OFFSET, entry_rel + 4);
145   outw(ctx - I0_OFFSET, arg1);
146   outw(ctx - I1_OFFSET, arg2);
147
148 #ifdef POK_NEEDS_DEBUG
149   printf ("space_context_create part_id=%d entry=%x stack=%x arg1=%x arg2=%x\n",
150           id, entry_rel, stack_rel, arg1, arg2);
151 #endif
152
153   return SPARC_PARTITION_BASE_VADDR + stack_rel - 0x40;
154 }
```

4.20.3.8 pok_ret_t pok_space_switch (uint8_t *old_partition_id*, uint8_t *new_partition_id*)

Switch adress space in MMU (context register).

Definition at line 108 of file space.c.

```
110 {
111   (void) old_partition_id;
112
113   asm volatile ("flush\n"
114                 "sta %0, [%1] %2;\n"
115                 : /* no output */
116                 : "r" (new_partition_id), "r" (MMU_CTX_REG), "i" (
    ASI_M_MMUREGS)
117                 : "memory");
118   return (POK_ERRNO_OK);
119 }
```

4.20.4 Variable Documentation

4.20.4.1 struct pok_space spaces[POK_CONFIG_NB_PARTITIONS]

Definition at line 47 of file space.c.

4.21 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.c File Reference

Handle address spaces.

```
#include <types.h>
#include <errno.h>
#include <libc.h>
#include <bsp.h>
#include <arch.h>
#include <arch/x86/interrupt.h>
#include "gdt.h"
#include "tss.h"
#include "space.h"
```

Macros

- #define KERNEL_STACK_SIZE 8192

Functions

- pok_ret_t pok_create_space (uint8_t partition_id, uint32_t addr, uint32_t size)

- pok_ret_t pok_space_switch (uint8_t old_partition_id, uint8_t new_partition_id)
- uint32_t pok_space_base_vaddr (uint32_t addr)
- uint32_t pok_space_context_create (uint8_t partition_id, uint32_t entry_rel, uint32_t stack_rel, uint32_t arg1, uint32_t arg2)
- void pok_dispatch_space (uint8_t partition_id, uint32_t user_pc, uint32_t user_sp, uint32_t kernel_sp, uint32_t arg1, uint32_t arg2)

4.21.1 Detailed Description

Handle address spaces.

Author

Julian Pidancet

Definition in file space.c.

4.21.2 Macro Definition Documentation

4.21.2.1 #define KERNEL_STACK_SIZE 8192

Definition at line 38 of file space.c.

4.21.3 Function Documentation

4.21.3.1 pok_ret_t pok_create_space (uint8_t *partition_id*, uint32_t *addr*, uint32_t *size*)

Set ptd and pte for the given partition.

Definition at line 40 of file space.c.

```
43 {
44    gdt_set_segment (GDT_PARTITION_CODE_SEGMENT (partition_id),
45          addr, size, GDTE_CODE, 3);
46
47    gdt_set_segment (GDT_PARTITION_DATA_SEGMENT (partition_id),
48          addr, size, GDTE_DATA, 3);
49
50    return (POK_ERRNO_OK);
51 }
```

4.21.3.2 void pok_dispatch_space (uint8_t *partition_id*, uint32_t *user_pc*, uint32_t *user_sp*, uint32_t *kernel_sp*, uint32_t *arg1*, uint32_t *arg2*)

Definition at line 114 of file space.c.

```
120 {
121    interrupt_frame   ctx;
122    uint32_t          code_sel;
123    uint32_t          data_sel;
124    uint32_t          sp;
125
126    code_sel = GDT_BUILD_SELECTOR (GDT_PARTITION_CODE_SEGMENT (
      partition_id), 0, 3);
127    data_sel = GDT_BUILD_SELECTOR (GDT_PARTITION_DATA_SEGMENT (
      partition_id), 0, 3);
128
129    sp = (uint32_t) &ctx;
130
131    memset (&ctx, 0, sizeof (interrupt_frame));
132
133    pok_arch_preempt_disable ();
```

```
134
135     ctx.es = ctx.ds = ctx.ss = data_sel;
136
137     ctx.__esp    = (uint32_t) (&ctx.error); /* for pusha */
138     ctx.eip      = user_pc;
139     ctx.eax      = arg1;
140     ctx.ebx      = arg2;
141     ctx.cs       = code_sel;
142     ctx.eflags   = 1 << 9;
143     ctx.esp      = user_sp;
144
145     tss_set_esp0 (kernel_sp);
146
147     asm ("mov %0, %%esp          \n"
148          "pop %%es               \n"
149          "pop %%ds               \n"
150          "popa                   \n"
151          "addl $4, %%esp         \n"
152          "iret                   \n"
153          :
154          : "m" (sp)
155        );
156 }
```

4.21.3.3 uint32_t pok_space_base_vaddr (uint32_t *addr*)

Returns

partition virtual base adress.

See Also

SPARC_PARTITION_BASE_VADDR

Definition at line 64 of file space.c.

```
65 {
66     (void) addr;
67     return (0);
68 }
```

4.21.3.4 uint32_t pok_space_context_create (uint8_t *partition_id*, uint32_t *entry_rel*, uint32_t *stack_rel*, uint32_t *arg1*, uint32_t *arg2*)

Create a new context in the given space

Initilize thread stack.

Definition at line 70 of file space.c.

```
75 {
76     char*             stack_addr;
77     space_context_t*  sp;
78
79     stack_addr = pok_bsp_mem_alloc (KERNEL_STACK_SIZE);
80
81     sp = (space_context_t *)
82        (stack_addr + KERNEL_STACK_SIZE - 4 - sizeof (
       space_context_t));
83
84     memset (sp, 0, sizeof (space_context_t));
85
86     sp->ctx.__esp  = (uint32_t)(&sp->ctx.eip); /* for pusha */
87     sp->ctx.eip    = (uint32_t)pok_dispatch_space;
88     sp->ctx.cs     = GDT_CORE_CODE_SEGMENT << 3;
89     sp->ctx.eflags = 1 << 9;
90
91     sp->arg1          = arg1;
92     sp->arg2          = arg2;
93     sp->kernel_sp     = (uint32_t)sp;
94     sp->user_sp       = stack_rel;
95     sp->user_pc       = entry_rel;
```

```
96    sp->partition_id  = partition_id;
97
98    return ((uint32_t) sp);
99 }
```

4.21.3.5 pok_ret_t pok_space_switch (uint8_t *old_partition_id*, uint8_t *new_partition_id*)

Switch from one space to another

Switch adress space in MMU (context register).

Definition at line 53 of file space.c.

```
55 {
56    gdt_disable (GDT_PARTITION_CODE_SEGMENT(old_partition_id));
57    gdt_disable (GDT_PARTITION_DATA_SEGMENT(old_partition_id));
58    gdt_enable (GDT_PARTITION_CODE_SEGMENT(new_partition_id));
59    gdt_enable (GDT_PARTITION_DATA_SEGMENT(new_partition_id));
60
61    return (POK_ERRNO_OK);
62 }
```

4.22 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/syscalls.c File Reference

```
#include <errno.h>
#include <core/debug.h>
#include <core/syscall.h>
#include <core/partition.h>
#include <types.h>
#include <libc.h>
```

Functions

- void pok_arch_sc_int (uint32_t num, uint32_t arg1, uint32_t arg2, uint32_t arg3, uint32_t arg4, uint32_t arg5)

4.22.1 Function Documentation

4.22.1.1 void pok_arch_sc_int (uint32_t *num*, uint32_t *arg1*, uint32_t *arg2*, uint32_t *arg3*, uint32_t *arg4*, uint32_t *arg5*)

Definition at line 26 of file syscalls.c.

```
28 {
29    uint8_t              part_id;
30
31    pok_syscall_info_t   syscall_info;
32    pok_syscall_args_t   syscall_args;
33    pok_syscall_id_t     syscall_id;
34
35    part_id  = pok_current_partition;
36
37    /* prepare syscall_info */
38    syscall_info.partition = part_id;
39    syscall_info.base_addr = pok_partitions[part_id].base_addr;
40    syscall_info.thread    = POK_SCHED_CURRENT_THREAD;
41
42    /* prepare syscall_args */
43    syscall_args.arg1 = arg1;
44    syscall_args.arg2 = arg2;
45    syscall_args.arg3 = arg3;
46    syscall_args.arg4 = arg4;
47    syscall_args.arg5 = arg5;
48
49    syscall_args.nargs = 5;
```

```
50
51    /* prepare syscall_id */
52    syscall_id = (pok_syscall_id_t) num;
53
54    if (POK_CHECK_PTR_IN_PARTITION(syscall_info.partition, &syscall_args) != 0)
55    {
56       /*
57        * Perform the syscall baby !
58        */
59      pok_core_syscall (syscall_id, &syscall_args, &syscall_info);
60    }
61 }
```

4.23 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/syscalls.c File Reference

Syscalls management in SPARC.

```
#include <errno.h>
#include <core/debug.h>
#include <core/syscall.h>
#include <core/partition.h>
#include <types.h>
#include <libc.h>
#include "thread.h"
#include "context_offset.h"
#include "traps.h"
#include "arch.h"
```

Functions

- void pok_arch_sc_int (void)
- void pok_syscalls_init (void)

4.23.1 Detailed Description

Syscalls management in SPARC.

Author

Fabien Chouteau

Definition in file syscalls.c.

4.23.2 Function Documentation

4.23.2.1 void pok_arch_sc_int (void)

Syscalls handler.

Definition at line 39 of file syscalls.c.

```
40 {
41   uint8_t *ctx = (uint8_t *)pok_arch_sp;
42   uint32_t num = *(uint32_t *)((char *)ctx - IO_OFFSET);
43   uint8_t              part_id;
44   pok_syscall_info_t   syscall_info;
45   pok_ret_t            syscall_ret;
46   pok_syscall_args_t   syscall_args;
47   pok_syscall_id_t     syscall_id;
48
49
```

```
50   part_id  = pok_current_partition;
51
52   /* prepare syscall_info */
53   syscall_info.partition = part_id;
54   syscall_info.base_addr = pok_partitions[part_id].base_addr;
55   syscall_info.thread    = POK_SCHED_CURRENT_THREAD;
56
57   /* prepare syscall_args */
58   syscall_args.arg1 = *(uint32_t *)(ctx - I1_OFFSET);
59   syscall_args.arg2 = *(uint32_t *)(ctx - I2_OFFSET);
60   syscall_args.arg3 = *(uint32_t *)(ctx - I3_OFFSET);
61   syscall_args.arg4 = *(uint32_t *)(ctx - I4_OFFSET);
62   syscall_args.arg5 = *(uint32_t *)(ctx - I5_OFFSET);
63
64   syscall_args.nargs = 5;
65
66   /* prepare syscall_id */
67   syscall_id = (pok_syscall_id_t) num;
68
69   /*
70    * No pointer check needed, syscall_args is allocated in kernel stack.
71    */
72   syscall_ret = pok_core_syscall (syscall_id, &syscall_args, &syscall_info);
73
74   *(uint32_t *)(ctx - I0_OFFSET) = syscall_ret;
75   *(uint32_t *)(ctx - PC_OFFSET) += 4; // skip "ta" instruction
76   *(uint32_t *)(ctx - NPC_OFFSET) += 4;
77 }
```

4.23.2.2 void pok_syscalls_init (void)

Syscalls initialization. Just register the syscall handler.

Definition at line 83 of file syscalls.c.

```
84 {
85   pok_arch_event_register(SPARC_TRAP_SYSCALL_BASE + 0x2,
      pok_arch_sc_int);
86 }
```

4.24 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/syscalls.c File Reference

This file implement system-calls for x86 platform.

```
#include <errno.h>
#include <core/debug.h>
#include <core/partition.h>
#include <core/syscall.h>
#include "gdt.h"
#include "event.h"
```

Macros

- #define PARTITION_ID(cs) (((cs >> 3) - 4) / 2)

Functions

- INTERRUPT_HANDLER_syscall (syscall_gate)
- pok_ret_t pok_syscall_init ()

4.24.1 Detailed Description

This file implement system-calls for x86 platform.

Author

Julian Pidancet
Julien Delange
Laurent Lec

Definition in file syscalls.c.

4.24.2 Macro Definition Documentation

4.24.2.1 #define PARTITION_ID(*cs*) (((cs >> 3) - 4) / 2)

Definition at line 33 of file syscalls.c.

4.24.3 Function Documentation

4.24.3.1 INTERRUPT_HANDLER_syscall (syscall_gate)

Definition at line 35 of file syscalls.c.

```
36 {
37    pok_syscall_info_t   syscall_info;
38    pok_ret_t            syscall_ret;
39    pok_syscall_args_t*  syscall_args;
40    pok_syscall_id_t     syscall_id;
41
42    /*
43     * Give informations about syscalls: which partition, thread
44     * initiates the syscall, the base addr of the partition and so on.
45     */
46    syscall_info.partition = PARTITION_ID (frame->cs);
47    syscall_info.base_addr = pok_partitions[syscall_info.partition].base_addr;
48    syscall_info.thread    = POK_SCHED_CURRENT_THREAD;
49
50    syscall_args = (pok_syscall_args_t*) (frame->ebx + syscall_info.
      base_addr);
51
52    /*
53     * Get the syscall id in the eax register
54     */
55    syscall_id = (pok_syscall_id_t) frame->eax;
56
57    /*
58     * Check that pointer is inside the adress space
59     */
60    if (POK_CHECK_PTR_IN_PARTITION(syscall_info.partition, syscall_args) == 0)
61    {
62          syscall_ret = POK_ERRNO_EINVAL;
63    }
64    else
65    {
66       /*
67        * Perform the syscall baby !
68        */
69       syscall_ret = pok_core_syscall (syscall_id, syscall_args, &syscall_info);
70    }
71
72    /*
73     * And finally, put the return value in eax register
74     */
75    asm ("movl %0, %%eax  \n"
76         :
77         : "m" (syscall_ret));
78 }
```

4.24.3.2 pok_ret_t pok_syscall_init ( )

Init system calls

Definition at line 83 of file syscalls.c.

```
84 {
85    pok_idt_set_gate (POK_SYSCALL_INT_NUMBER,
86                      GDT_CORE_CODE_SEGMENT << 3,
87                      (uint32_t) syscall_gate,
88                      IDTE_INTERRUPT,
89                      3);
90
91    return (POK_ERRNO_OK);
92 }
```

4.25 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.c File Reference

```
#include <bsp.h>
#include <libc.h>
#include <errno.h>
#include <core/thread.h>
#include "thread.h"
```

4.26 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/thread.c File Reference

Thread management.

```
#include <bsp.h>
#include <libc.h>
#include <errno.h>
#include <core/thread.h>
#include "thread.h"
#include "context_offset.h"
#include "ioports.h"
```

4.26.1 Detailed Description

Thread management.

Author

Fabien Chouteau

Definition in file thread.c.

4.27 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.c File Reference

```
#include <bsp.h>
#include <libc.h>
#include <errno.h>
#include <core/thread.h>
#include "gdt.h"
#include "thread.h"
```

4.28 /home/phipse/gsoc/pok/trunk/kernel/core/thread.c File Reference

Thread management in kernel.

```
#include <types.h>
#include <arch.h>
#include <core/debug.h>
#include <core/error.h>
#include <core/thread.h>
#include <core/sched.h>
#include <core/partition.h>
#include <core/time.h>
#include <core/instrumentation.h>
```

4.28.1 Detailed Description

Thread management in kernel.

Author

Julien Delange

Date

2008-2009

Definition in file thread.c.

4.29 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.h File Reference

```
#include <types.h>
```

Data Structures

- struct context_t
- struct volatile_context_t

Functions

- uint32_t pok_context_create (uint32_t id, uint32_t stack_size, uint32_t entry)
- void pok_context_switch (uint32_t *old_sp, uint32_t new_sp)

4.29.1 Function Documentation

4.29.1.1 uint32_t pok_context_create (uint32_t *id*, uint32_t *stack_size*, uint32_t *entry*)

4.29.1.2 void pok_context_switch (uint32_t * *old_sp*, uint32_t *new_sp*)

4.30 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/thread.h File Reference

```
#include <types.h>
```

Functions

- uint32_t pok_context_create (uint32_t id, uint32_t stack_size, uint32_t entry)
- void pok_context_switch (uint32_t *old_sp, uint32_t new_sp)

Variables

- uint32_t pok_arch_sp

4.30.1 Detailed Description

Author

Fabien Chouteau

Definition in file thread.h.

4.30.2 Function Documentation

4.30.2.1 uint32_t pok_context_create (uint32_t *id*, uint32_t *stack_size*, uint32_t *entry*)

4.30.2.2 void pok_context_switch (uint32_t * *old_sp*, uint32_t *new_sp*)

4.30.3 Variable Documentation

4.30.3.1 uint32_t pok_arch_sp

4.31 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.h File Reference

```
#include <types.h>
```

Data Structures

- struct context_t
- struct start_context_t

Functions

- uint32_t pok_context_create (uint32_t id, uint32_t stack_size, uint32_t entry)
- void pok_context_switch (uint32_t *old_sp, uint32_t new_sp)
- void pok_context_reset (uint32_t stack_size, uint32_t stack_addr)

4.31.1 Function Documentation

4.31.1.1 uint32_t pok_context_create (uint32_t *id*, uint32_t *stack_size*, uint32_t *entry*)

4.31.1.2 void pok_context_reset (uint32_t *stack_size*, uint32_t *stack_addr*)

4.31.1.3 void pok_context_switch (uint32_t * *old_sp*, uint32_t *new_sp*)

4.32 /home/phipse/gsoc/pok/trunk/kernel/include/core/thread.h File Reference

4.33 /home/phipse/gsoc/pok/trunk/kernel/arch/ppc/timer.c File Reference

```
#include <errno.h>
#include <bsp.h>
#include <core/time.h>
#include <core/sched.h>
```

Macros

- #define BUS_FREQ (100 ∗ 1000000U)
- #define FREQ_DIV 40

Functions

- void pok_arch_decr_int (void)
- pok_ret_t pok_bsp_time_init ()

4.33.1 Macro Definition Documentation

4.33.1.1 #define BUS_FREQ (100 ∗ 1000000U)

Definition at line 24 of file timer.c.

4.33.1.2 #define FREQ_DIV 40

Definition at line 26 of file timer.c.

4.33.2 Function Documentation

4.33.2.1 void pok_arch_decr_int (void)

Definition at line 73 of file timer.c.

```
74 {
75   int err;
76
77   do
78   {
79     err = pok_arch_set_decr();
80
81     pok_tick_counter += FREQ_DIV;
82   } while (err != POK_ERRNO_OK);
83
84   pok_sched ();
85 }
```

4.33.2.2 pok_ret_t pok_bsp_time_init ( )

Definition at line 87 of file timer.c.

```
88 {
89   time_inter = (BUS_FREQ * FREQ_DIV) / POK_TIMER_FREQUENCY;
90   time_last = get_ppc_tb ();
91   pok_arch_set_decr();
92
93   return (POK_ERRNO_OK);
94 }
```

4.34 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/timer.c File Reference

Leon3 timer management.

```
#include <errno.h>
#include <bsp.h>
#include <core/time.h>
#include <core/sched.h>
#include <arch.h>
#include "ioports.h"
#include "sparc_conf.h"
#include "timer.h"
#include "irq.h"
#include "../traps.h"
```

Functions

- void timer_isr (void)
- pok_ret_t pok_bsp_time_init ()

4.34.1 Detailed Description

Leon3 timer management.

Author

Fabien Chouteau

Definition in file timer.c.

4.34.2 Function Documentation

4.34.2.1 pok_ret_t pok_bsp_time_init ( )

Initialize the timer, register the ISR and unmask the interrupt.

See Also

unmask_irq(irq_nbr)

Definition at line 50 of file timer.c.

```
51 {
52   outw(TIMER1 + TIMER_SCALER_OFFSET, 1);
53   outw(TIMER1 + TIMER_SCAL_RELOAD_OFFSET, 1);
54
55   outw(TIMER1 + TIMER_CNT_VAL_OFFSET, 1);
56   outw(TIMER1 + TIMER_RELOAD_OFFSET, SPARC_PROC_FREQ /
      POK_TIMER_FREQUENCY);
57   outw(TIMER1 + TIMER_CTRL_OFFSET,
```

```
58        TIMER_CTRL_EN | TIMER_CTRL_RS | TIMER_CTRL_LD |
      TIMER_CTRL_IE);
59
60    pok_arch_event_register(SPARC_TRAP_IRQ_BASE +
      TIMER_IRQ, timer_isr);
61    unmask_irq(TIMER_IRQ);
62    return (POK_ERRNO_OK);
63 }
```

4.34.2.2 void timer_isr (void)

Timer interrupt subroutine.

See Also

ack_irq(irq_nbr)

Definition at line 39 of file timer.c.

```
40 {
41    ack_irq(TIMER_IRQ);
42    CLOCK_HANDLER
43    return;
44 }
```

4.35 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/context_offset.h File Reference

Define registers offset in context stack.

Macros

- #define L0_OFFSET 0x00
- #define L1_OFFSET 0x04
- #define L2_OFFSET 0x08
- #define L3_OFFSET 0x0c
- #define L4_OFFSET 0x10
- #define L5_OFFSET 0x14
- #define L6_OFFSET 0x18
- #define L7_OFFSET 0x1c
- #define I0_OFFSET 0x20
- #define I1_OFFSET 0x24
- #define I2_OFFSET 0x28
- #define I3_OFFSET 0x2c
- #define I4_OFFSET 0x30
- #define I5_OFFSET 0x34
- #define I6_OFFSET 0x38
- #define I7_OFFSET 0x3c
- #define G7_OFFSET 0x04
- #define G6_OFFSET 0x08
- #define G5_OFFSET 0x0c
- #define G4_OFFSET 0x10
- #define G3_OFFSET 0x14
- #define G2_OFFSET 0x18
- #define G1_OFFSET 0x1c
- #define WIM_OFFSET 0x40
- #define PSR_OFFSET 0x44
- #define Y_OFFSET 0x48
- #define PC_OFFSET 0x4c
- #define NPC_OFFSET 0x50
- #define RESTORE_CNT_OFFSET 0x54

4.35.1 Detailed Description

Define registers offset in context stack.

Author

Fabien Chouteau

Definition in file context_offset.h.

4.35.2 Macro Definition Documentation

4.35.2.1 #define G1_OFFSET 0x1c

Definition at line 57 of file context_offset.h.

4.35.2.2 #define G2_OFFSET 0x18

Definition at line 56 of file context_offset.h.

4.35.2.3 #define G3_OFFSET 0x14

Definition at line 55 of file context_offset.h.

4.35.2.4 #define G4_OFFSET 0x10

Definition at line 54 of file context_offset.h.

4.35.2.5 #define G5_OFFSET 0x0c

Definition at line 53 of file context_offset.h.

4.35.2.6 #define G6_OFFSET 0x08

Definition at line 52 of file context_offset.h.

4.35.2.7 #define G7_OFFSET 0x04

Definition at line 51 of file context_offset.h.

4.35.2.8 #define I0_OFFSET 0x20

Definition at line 38 of file context_offset.h.

4.35.2.9 #define I1_OFFSET 0x24

Definition at line 39 of file context_offset.h.

4.35.2.10 #define I2_OFFSET 0x28

Definition at line 40 of file context_offset.h.

4.35.2.11 #define I3_OFFSET 0x2c

Definition at line 41 of file context_offset.h.

4.35.2.12 #define I4_OFFSET 0x30

Definition at line 42 of file context_offset.h.

4.35.2.13 #define I5_OFFSET 0x34

Definition at line 43 of file context_offset.h.

4.35.2.14 #define I6_OFFSET 0x38

Definition at line 44 of file context_offset.h.

4.35.2.15 #define I7_OFFSET 0x3c

Definition at line 45 of file context_offset.h.

4.35.2.16 #define L0_OFFSET 0x00

Definition at line 30 of file context_offset.h.

4.35.2.17 #define L1_OFFSET 0x04

Definition at line 31 of file context_offset.h.

4.35.2.18 #define L2_OFFSET 0x08

Definition at line 32 of file context_offset.h.

4.35.2.19 #define L3_OFFSET 0x0c

Definition at line 33 of file context_offset.h.

4.35.2.20 #define L4_OFFSET 0x10

Definition at line 34 of file context_offset.h.

4.35.2.21 #define L5_OFFSET 0x14

Definition at line 35 of file context_offset.h.

4.35.2.22 #define L6_OFFSET 0x18

Definition at line 36 of file context_offset.h.

4.35.2.23 #define L7_OFFSET 0x1c

Definition at line 37 of file context_offset.h.

4.35.2.24 #define NPC_OFFSET 0x50

Definition at line 66 of file context_offset.h.

4.35.2.25 #define PC_OFFSET 0x4c

Definition at line 65 of file context_offset.h.

4.35.2.26 #define PSR_OFFSET 0x44

Definition at line 63 of file context_offset.h.

4.35.2.27 #define RESTORE_CNT_OFFSET 0x54

Definition at line 67 of file context_offset.h.

4.35.2.28 #define WIM_OFFSET 0x40

Definition at line 62 of file context_offset.h.

4.35.2.29 #define Y_OFFSET 0x48

Definition at line 64 of file context_offset.h.

4.36 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/irq.h File Reference

Leon3 IRQ management.

```
#include "ioports.h"
```

Macros

- #define IRQMP_BASE 0x80000200
- #define IRQMP_CLEAR_OFFSET 0x10U
- #define IRQMP_MASK0_OFFSET 0x40U
- #define unmask_irq(irq_nbr)
- #define ack_irq(irq_nbr) outw(IRQMP_BASE + IRQMP_CLEAR_OFFSET, (1 << (irq_nbr)))

4.36.1 Detailed Description

Leon3 IRQ management.

Author

Fabien Chouteau

Definition in file irq.h.

4.36.2 Macro Definition Documentation

4.36.2.1 #define ack_irq(*irq_nbr*) outw(IRQMP_BASE + IRQMP_CLEAR_OFFSET, (1 << (irq_nbr)))

Acknowledge the given irq.

Definition at line 44 of file irq.h.

4.36.2.2 #define IRQMP_BASE 0x80000200

Leon3 IRQMP IO adress

Definition at line 28 of file irq.h.

4.36.2.3 #define IRQMP_CLEAR_OFFSET 0x10U

Clear register offset

Definition at line 30 of file irq.h.

4.36.2.4 #define IRQMP_MASK0_OFFSET 0x40U

Mask register offset

Definition at line 31 of file irq.h.

4.36.2.5 #define unmask_irq(*irq_nbr*)

Value:

```
outw(IRQMP_BASE + IRQMP_MASK0_OFFSET,       \
                              inb(IRQMP_BASE +
      IRQMP_MASK0_OFFSET) | (1 << (irq_nbr)))
```

Unmask the given irq.

Definition at line 37 of file irq.h.

4.37 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/sparc_conf.h File Reference

Define all constant values for a SPARC bsp.

Macros

- #define SPARC_RAM_ADDR 0x40000000
- #define SPARC_PROC_FREQ 50000000U
- #define WINDOWS_NBR 8
- #define ASI_MMU_BYPASS 0x1c /* not sparc v8 compliant */
- #define SPARC_PAGE_SIZE (256 * 1024)
- #define SPARC_PARTITION_SIZE SPARC_PAGE_SIZE
- #define SPARC_PARTITION_BASE_VADDR 0x0

4.37.1 Detailed Description

Define all constant values for a SPARC bsp.

Author

Fabien Chouteau

Definition in file sparc_conf.h.

4.37.2 Macro Definition Documentation

4.37.2.1 #define ASI_MMU_BYPASS 0x1c /* not sparc v8 compliant */

Definition at line 32 of file sparc_conf.h.

4.37.2.2 #define SPARC_PAGE_SIZE (256 * 1024)

Page size (256 Kbytes)

Definition at line 34 of file sparc_conf.h.

4.37.2.3 #define SPARC_PARTITION_BASE_VADDR 0x0

Partition virtual base adress. Should always be 0x0

Definition at line 37 of file sparc_conf.h.

4.37.2.4 #define SPARC_PARTITION_SIZE SPARC_PAGE_SIZE

Maximum partition size

Definition at line 35 of file sparc_conf.h.

4.37.2.5 #define SPARC_PROC_FREQ 50000000U

Processor frequency (in Hz)

Definition at line 28 of file sparc_conf.h.

4.37.2.6 #define SPARC_RAM_ADDR 0x40000000

RAM base adress

Definition at line 26 of file sparc_conf.h.

4.37.2.7 #define WINDOWS_NBR 8

Number of register windows

Definition at line 30 of file sparc_conf.h.

4.38 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/timer.h File Reference

Macros

- #define TIMER_CTRL_EN (1 << 0)
- #define TIMER_CTRL_RS (1 << 1)
- #define TIMER_CTRL_LD (1 << 2)
- #define TIMER_CTRL_IE (1 << 3)
- #define TIMER_CTRL_IP (1 << 4)
- #define TIMER_CTRL_CH (1 << 5)
- #define TIMER_CTRL_DH (1 << 6)
- #define TIMER_SCALER_OFFSET 0x00
- #define TIMER_SCAL_RELOAD_OFFSET 0x04
- #define TIMER_CNT_VAL_OFFSET 0x10
- #define TIMER_RELOAD_OFFSET 0x14
- #define TIMER_CTRL_OFFSET 0x18
- #define TIMER_IRQ 0x8U
- #define TIMER1 0x80000300

4.38.1 Detailed Description

Author

Fabien Chouteau

Definition in file timer.h.

4.38.2 Macro Definition Documentation

4.38.2.1 #define TIMER1 0x80000300

first Leon3 TIMER IO adress

Definition at line 42 of file timer.h.

4.38.2.2 #define TIMER_CNT_VAL_OFFSET 0x10

Counter value register offset

Definition at line 36 of file timer.h.

4.38.2.3 #define TIMER_CTRL_CH (1 << 5)

Chain

Definition at line 30 of file timer.h.

4.38.2.4 #define TIMER_CTRL_DH (1 << 6)

Debug Halt

Definition at line 31 of file timer.h.

4.38.2.5 #define TIMER_CTRL_EN (1 << 0)

Enable

Definition at line 25 of file timer.h.

4.38.2.6 #define TIMER_CTRL_IE (1 << 3)

Interrupt enable

Definition at line 28 of file timer.h.

4.38.2.7 #define TIMER_CTRL_IP (1 << 4)

Interrupt Pending

Definition at line 29 of file timer.h.

4.38.2.8 #define TIMER_CTRL_LD (1 << 2)

Load

Definition at line 27 of file timer.h.

4.38.2.9 #define TIMER_CTRL_OFFSET 0x18

Control register offset

Definition at line 38 of file timer.h.

4.38.2.10 #define TIMER_CTRL_RS (1 << 1)

Restart

Definition at line 26 of file timer.h.

4.38.2.11 #define TIMER_IRQ 0x8U

Definition at line 40 of file timer.h.

4.38.2.12 #define TIMER_RELOAD_OFFSET 0x14

Counter reload register offset

Definition at line 37 of file timer.h.

4.38.2.13 #define TIMER_SCAL_RELOAD_OFFSET 0x04

Scaler reload register offset

Definition at line 34 of file timer.h.

4.38.2.14 #define TIMER_SCALER_OFFSET 0x00

Scaler value register offset

Definition at line 33 of file timer.h.

4.39 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/psr.h File Reference

Processor State Register utils.

Macros

- #define PSR_ET 0x20
- #define PSR_PS 0x40
- #define PSR_S 0x80
- #define PSR_CWP_MASK 0x1F
- #define PSR_PIL(pil) (((pil) & 0xF) << 8)

4.39.1 Detailed Description

Processor State Register utils.

Author

Fabien Chouteau

Definition in file psr.h.

4.39.2 Macro Definition Documentation

4.39.2.1 #define PSR_CWP_MASK 0x1F

Current Window Pointer Mask

Definition at line 29 of file psr.h.

4.39.2.2 #define PSR_ET 0x20

enable traps

Definition at line 26 of file psr.h.

4.39.2.3 #define PSR_PIL(*pil*) (((pil) & 0xF) << 8)

Proc Interrupt Level

Definition at line 30 of file psr.h.

4.39.2.4 #define PSR_PS 0x40

previous supervisor

Definition at line 27 of file psr.h.

4.39.2.5 #define PSR_S 0x80

supervisor

Definition at line 28 of file psr.h.

4.40 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.h File Reference

```
#include <types.h>
```

Macros

- #define LEON_CTX_NBR 256

PTD/PTE ET field

(cf SPARC V8 Manual, page 247)

- #define MM_ET_INVALID 0x0
- #define MM_ET_PTD 0x1
- #define MM_ET_PTE 0x2

PTE ACC field

Acces permisions. (cf SPARC V8 Manual, page 248)

- #define MM_ACC_R (0x0 << 2)
- #define MM_ACC_RW (0x1 << 2)
- #define MM_ACC_RE (0x2 << 2)
- #define MM_ACC_RWE (0x3 << 2)
- #define MM_ACC_E (0x4 << 2)
- #define MM_ACC_R_S_RW (0x5 << 2)
- #define MM_ACC_S_RE (0x6 << 2)
- #define MM_ACC_S_RWE (0x7 << 2)

PTE misc fields

(cf SPARC V8 Manual, page 248)

- #define MM_CACHEABLE (1 << 7)
- #define MM_MODIFIED (1 << 6)
- #define MM_REFERENCED (1 << 5)

MMU levels utils

- #define MM_LVL1_ENTRIES_NBR 256
- #define MM_LVL1_PAGE_SIZE (64 * 64 * 4 * 1024)
- #define mm_index1(addr) (((addr) >> 24) & 0xFF)
- #define MM_LVL2_ENTRIES_NBR 64
- #define MM_LVL2_PAGE_SIZE (64 * 4 * 1024)
- #define mm_index2(addr) (((addr) >> 18) & 0x3F)
- #define MM_LVL3_ENTRIES_NBR 64
- #define MM_LVL3_PAGE_SIZE (4 * 1024)
- #define mm_index3(addr) (((addr) >> 12) & 0x3F)

MMU ASI and registers

- #define ASI_M_MMUREGS 0x19 /* not sparc v8 compliant */
- #define MMU_CTRL_REG 0x00000000
- #define MMU_CTXTBL_PTR 0x00000100
- #define MMU_CTX_REG 0x00000200
- #define MMU_FAULT_STATUS 0x00000300
- #define MMU_FAULT_ADDR 0x00000400

Typedefs

- typedef uint32_t pte
- typedef uint32_t ptd

Functions

- void pok_arch_space_init (void)

4.40.1 Detailed Description

Author

Fabien Chouteau

Definition in file space.h.

4.40.2 Macro Definition Documentation

4.40.2.1 #define ASI_M_MMUREGS 0x19 /* not sparc v8 compliant */

Definition at line 97 of file space.h.

4.40.2.2 #define LEON_CTX_NBR 256

Maximum number of contexts

Definition at line 105 of file space.h.

4.40.2.3 #define MM_ACC_E (0x4 << 2)

All Execute only

Definition at line 46 of file space.h.

4.40.2.4 #define MM_ACC_R (0x0 << 2)

All Read only

Definition at line 42 of file space.h.

4.40.2.5 #define MM_ACC_R_S_RW (0x5 << 2)

User Read only, Supervisor Read Write

Definition at line 47 of file space.h.

4.40.2.6 #define MM_ACC_RE (0x2 << 2)

All Read Execute

Definition at line 44 of file space.h.

4.40.2.7 #define MM_ACC_RW (0x1 << 2)

All Read Write

Definition at line 43 of file space.h.

4.40.2.8 #define MM_ACC_RWE (0x3 << 2)

All Read Write Execute

Definition at line 45 of file space.h.

4.40.2.9 #define MM_ACC_S_RE (0x6 << 2)

Supervisor Read Write Execute

Definition at line 49 of file space.h.

4.40.2.10 #define MM_ACC_S_RWE (0x7 << 2)

Supervisor Read Execute

Definition at line 50 of file space.h.

4.40.2.11 #define MM_CACHEABLE (1 << 7)

Definition at line 58 of file space.h.

4.40.2.12 #define MM_ET_INVALID 0x0

Invalid

Definition at line 32 of file space.h.

4.40.2.13 #define MM_ET_PTD 0x1

Page Table Descriptor

Definition at line 33 of file space.h.

4.40.2.14 #define MM_ET_PTE 0x2

Page Table Entry

Definition at line 34 of file space.h.

4.40.2.15 #define mm_index1(*addr*) (((addr) >> 24) & 0xFF)

Compute the index in 1st level table for the given adress.

Definition at line 73 of file space.h.

4.40.2.16 #define mm_index2(*addr*) (((addr) >> 18) & 0x3F)

Compute the index in 2nd level table for the given adress.

Definition at line 81 of file space.h.

4.40.2.17 #define mm_index3(*addr*) (((addr) >> 12) & 0x3F)

Compute the index in 3rd level table for the given adress.

Definition at line 89 of file space.h.

4.40.2.18 #define MM_LVL1_ENTRIES_NBR 256

Number of entries in 1st level table

Definition at line 67 of file space.h.

4.40.2.19 #define MM_LVL1_PAGE_SIZE (64 * 64 * 4 * 1024)

16 MegaBytes

Definition at line 68 of file space.h.

4.40.2.20 #define MM_LVL2_ENTRIES_NBR 64

Number of entries in 2nd level table

Definition at line 75 of file space.h.

4.40.2.21 #define MM_LVL2_PAGE_SIZE (64 * 4 * 1024)

256 KiloBytes

Definition at line 76 of file space.h.

4.40.2.22 #define MM_LVL3_ENTRIES_NBR 64

Number of entries in 3rd level table

Definition at line 83 of file space.h.

4.40.2.23 #define MM_LVL3_PAGE_SIZE (4 * 1024)

4 KiloBytes

Definition at line 84 of file space.h.

4.40.2.24 #define MM_MODIFIED (1 << 6)

Definition at line 59 of file space.h.

4.40.2.25 #define MM_REFERENCED (1 << 5)

Definition at line 60 of file space.h.

4.40.2.26 #define MMU_CTRL_REG 0x00000000

Definition at line 98 of file space.h.

4.40.2.27 #define MMU_CTX_REG 0x00000200

Definition at line 100 of file space.h.

4.40.2.28 #define MMU_CTXTBL_PTR 0x00000100

Definition at line 99 of file space.h.

4.40.2.29 #define MMU_FAULT_ADDR 0x00000400

Definition at line 102 of file space.h.

4.40.2.30 #define MMU_FAULT_STATUS 0x00000300

Definition at line 101 of file space.h.

4.40.3 Typedef Documentation

4.40.3.1 typedef uint32_t ptd

Definition at line 108 of file space.h.

4.40.3.2 typedef uint32_t pte

Definition at line 107 of file space.h.

4.40.4 Function Documentation

4.40.4.1 void pok_arch_space_init (void)

Initilize MMU tables.

Definition at line 132 of file space.c.

```
133 {
134   uint32_t sdr1;
135
136   pt_base = 0;
137   pt_mask = 0x3ff;
138
139   sdr1 = pt_base | (pt_mask >> 10);
140   asm volatile ("mtsdr1 %0" : : "r"(sdr1));
141 }
```

4.41 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.h File Reference

```
#include <types.h>
#include "thread.h"
```

Data Structures

- struct space_context_t

4.41.1 Detailed Description

Author

Julian Pidancet

Date

2008-2009

Definition in file space.h.

4.42 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/syscalls.h File Reference

Functions

- void pok_syscalls_init (void)

4.42.1 Detailed Description

Author

Fabien Chouteau

Definition in file syscalls.h.

4.42.2 Function Documentation

4.42.2.1 void pok_syscalls_init (void)

Syscalls initialization. Just register the syscall handler.

Definition at line 83 of file syscalls.c.

```
84 {
85   pok_arch_event_register(SPARC_TRAP_SYSCALL_BASE + 0x2,
      pok_arch_sc_int);
86 }
```

4.43 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/traps.c File Reference

Traps management.

```
#include <types.h>
#include <errno.h>
#include <libc.h>
#include <core/debug.h>
#include "thread.h"
#include "traps.h"
```

Functions

- pok_ret_t traps_init (void)
- void trap_handler (unsigned int pc, unsigned int npc, unsigned int psr, unsigned int trap_nb, unsigned int restore_counter, unsigned int stack_pointer)

Variables

- sparc_traps_handler pok_sparc_isr [256]

4.43.1 Detailed Description

Traps management.

Author

Fabien Chouteau

Definition in file traps.c.

4.43.2 Function Documentation

4.43.2.1 void trap_handler (unsigned int *pc,* unsigned int *npc,* unsigned int *psr,* unsigned int *trap_nb,* unsigned int *restore_counter,* unsigned int *stack_pointer*)

Function called by interrupt pre-handler. Call the correct handler for the given trap number.

Parameters

trap_nb	The number of the current trap. (cf SPARC V8 Manual, page 76)
stack_pointer	Adress of the interrupted stack.

See Also

pok_arch_sp

Definition at line 53 of file traps.c.

```
59 {
60   (void)restore_counter;
61
62   pok_arch_sp = stack_pointer;
63
64   if (pok_sparc_isr[trap_nb] != NULL)
65   {
66     pok_sparc_isr[trap_nb]();
67   }
68   else
69   {
70 #ifdef POK_NEEDS_DEBUG
71     printf ("[KERNEL] [ERROR] Unhandled trap: 0x%x %%PSR=%x %%PC=%x %%nPC=%x %%sp=0x%x\n", trap_nb, psr, pc
     , npc, stack_pointer);
72     printf("%%psr : impl:0x%x ver:%x nzvc:%u%u%u%u EC:%u EF:%u PIL:0x%x S:%u PS:%u ET:%u CWP:%u\n\r",
73            (psr >> 28) & 0xF, (psr >> 24) & 0xF,
74            (psr >> 23) & 0x1, (psr >> 22) & 0x1c, (psr >> 21) & 0x1, (psr >> 20) & 0x1,
75            (psr >> 23) & 0x1, (psr >> 12) & 0x1, (psr >> 8) & 0xF, (psr >> 7) & 0x1, (psr >> 6) & 0x1,
76            (psr >> 5) & 0x1, psr & 0xF);
77 #else
78     (void)psr;
79     (void)npc;
80     (void)pc;
81 #endif
82     POK_FATAL ("Unhandled trap");
83   }
84   return;
85 }
```

4.43.2.2 pok_ret_t traps_init (void)

Initialize ISR table.

See Also

pok_sparc_isr

Definition at line 40 of file traps.c.

```
41 {
42   memset((unsigned char *)pok_sparc_isr, 0x0, sizeof (
      pok_sparc_isr));
43   return POK_ERRNO_OK;
44 }
```

4.43.3 Variable Documentation

4.43.3.1 sparc_traps_handler pok_sparc_isr[256]

Interrupt subroutine table.

Definition at line 34 of file traps.c.

4.44 /home/phipse/gsoc/pok/trunk/kernel/arch/sparc/traps.h File Reference

```
#include <types.h>
#include <errno.h>
```

Macros

- #define SPARC_TRAP_IRQ_BASE 0x10
- #define SPARC_TRAP_SYSCALL_BASE 0x80

Typedefs

- typedef void(* sparc_traps_handler)(void)

Functions

- pok_ret_t traps_init (void)

Variables

- sparc_traps_handler pok_sparc_isr [256]

4.44.1 Detailed Description

Author

Fabien Chouteau

Definition in file traps.h.

4.44.2 Macro Definition Documentation

4.44.2.1 #define SPARC_TRAP_IRQ_BASE 0x10

Definition at line 28 of file traps.h.

4.44.2.2 #define SPARC_TRAP_SYSCALL_BASE 0x80

Definition at line 29 of file traps.h.

4.44.3 Typedef Documentation

4.44.3.1 typedef void(* sparc_traps_handler)(void)

Definition at line 31 of file traps.h.

4.44.4 Function Documentation

4.44.4.1 pok_ret_t traps_init (void)

Initialize ISR table.

See Also

pok_sparc_isr

Definition at line 40 of file traps.c.

```
41 {
42   memset((unsigned char *)pok_sparc_isr, 0x0, sizeof (
      pok_sparc_isr));
43   return POK_ERRNO_OK;
44 }
```

4.44.5 Variable Documentation

4.44.5.1 sparc_traps_handler pok_sparc_isr[256]

Interrupt subroutine table.

Definition at line 34 of file traps.c.

4.45 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.c File Reference

```
#include <libc.h>
#include <types.h>
#include <errno.h>
#include <core/syscall.h>
#include "event.h"
#include "sysdesc.h"
```

Macros

- #define IDT_SIZE 256

Functions

- pok_ret_t pok_event_init ()
- pok_ret_t pok_idt_init ()
- void pok_idt_set_gate (uint16_t index, uint16_t segsel, uint32_t offset, e_idte_type t, int dpl)

Variables

- idt_entry_t pok_idt [IDT_SIZE]

4.45.1 Macro Definition Documentation

4.45.1.1 #define IDT_SIZE 256

Definition at line 27 of file event.c.

4.45.2 Function Documentation

4.45.2.1 pok_ret_t pok_event_init ( )

Definition at line 31 of file event.c.

```
32 {
33    pok_idt_init ();
34
35 #if defined (POK_NEEDS_DEBUG) || defined (POK_NEEDS_ERROR_HANDLING)
36    pok_exception_init ();
37 #endif
38
39    pok_syscall_init ();
40
41    return (POK_ERRNO_OK);
42 }
```

4.45.2.2 pok_ret_t pok_idt_init ( )

Definition at line 44 of file event.c.

```
45 {
46    sysdesc_t sysdesc;
47
48    /* Clear table */
49    memset(pok_idt, 0, sizeof (idt_entry_t) * IDT_SIZE);
50
51    /* Load IDT */
52    sysdesc.limit = sizeof (pok_idt);
53    sysdesc.base = (uint32_t)pok_idt;
54
55    asm ("lidt %0"
56         :
57         : "m" (sysdesc));
58
59   return (POK_ERRNO_OK);
60 }
```

4.45.2.3 void pok_idt_set_gate (uint16_t *index*, uint16_t *segsel*, uint32_t *offset*, e_idte_type *t*, int *dpl*)

Definition at line 62 of file event.c.

```
67 {
68    pok_idt[index].offset_low   = (offset) & 0xFFFF;
69    pok_idt[index].offset_high  = (offset >> 16) & 0xFFFF;
70    pok_idt[index].segsel       = segsel;
71    pok_idt[index].dpl          = dpl;
72    pok_idt[index].type         = t;
73    pok_idt[index].d            = 1;
74    pok_idt[index].res0         = 0; /* reserved */
75    pok_idt[index].res1         = 0; /* reserved */
76    pok_idt[index].present   = 1;
77 }
```

4.45.3 Variable Documentation

4.45.3.1 idt_entry_t pok_idt[IDT_SIZE]

Definition at line 29 of file event.c.

4.46 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.h File Reference

```
#include <types.h>
#include <arch/x86/interrupt.h>
#include "gdt.h"
```

Data Structures

- struct __attribute__

Macros

- #define EXCEPTION_DIVIDE_ERROR 0
- #define EXCEPTION_DEBUG 1
- #define EXCEPTION_NMI 2
- #define EXCEPTION_BREAKPOINT 3
- #define EXCEPTION_OVERFLOW 4
- #define EXCEPTION_BOUNDRANGE 5
- #define EXCEPTION_INVALIDOPCODE 6
- #define EXCEPTION_NOMATH_COPROC 7
- #define EXCEPTION_DOUBLEFAULT 8
- #define EXCEPTION_COPSEG_OVERRUN 9
- #define EXCEPTION_INVALID_TSS 10
- #define EXCEPTION_SEGMENT_NOT_PRESENT 11
- #define EXCEPTION_STACKSEG_FAULT 12
- #define EXCEPTION_GENERAL_PROTECTION 13
- #define EXCEPTION_PAGEFAULT 14
- #define EXCEPTION_RESERVED 15
- #define EXCEPTION_FPU_FAULT 16
- #define EXCEPTION_ALIGNEMENT_CHECK 17
- #define EXCEPTION_MACHINE_CHECK 18
- #define EXCEPTION_SIMD_FAULT 19

Typedefs

- typedef enum e_idte_type e_idte_type

Enumerations

- enum e_idte_type { IDTE_TASK = 5, IDTE_INTERRUPT = 6, IDTE_TRAP = 7 }

Functions

- void pok_idt_set_gate (uint16_t index, uint16_t segsel, uint32_t offset, e_idte_type t, int dpl)
- pok_ret_t pok_idt_init ()
- pok_ret_t pok_exception_init ()
- pok_ret_t pok_event_init ()
- pok_ret_t pok_syscall_init ()

4.46.1 Macro Definition Documentation

4.46.1.1 #define EXCEPTION_ALIGNEMENT_CHECK 17

Definition at line 63 of file event.h.

4.46.1.2 #define EXCEPTION_BOUNDRANGE 5

Definition at line 51 of file event.h.

4.46.1.3 #define EXCEPTION_BREAKPOINT 3

Definition at line 49 of file event.h.

4.46.1.4 #define EXCEPTION_COPSEG_OVERRUN 9

Definition at line 55 of file event.h.

4.46.1.5 #define EXCEPTION_DEBUG 1

Definition at line 47 of file event.h.

4.46.1.6 #define EXCEPTION_DIVIDE_ERROR 0

Definition at line 46 of file event.h.

4.46.1.7 #define EXCEPTION_DOUBLEFAULT 8

Definition at line 54 of file event.h.

4.46.1.8 #define EXCEPTION_FPU_FAULT 16

Definition at line 62 of file event.h.

4.46.1.9 #define EXCEPTION_GENERAL_PROTECTION 13

Definition at line 59 of file event.h.

4.46.1.10 #define EXCEPTION_INVALID_TSS 10

Definition at line 56 of file event.h.

4.46.1.11 #define EXCEPTION_INVALIDOPCODE 6

Definition at line 52 of file event.h.

4.46.1.12 #define EXCEPTION_MACHINE_CHECK 18

Definition at line 64 of file event.h.

4.46.1.13 #define EXCEPTION_NMI 2

Definition at line 48 of file event.h.

4.46.1.14 #define EXCEPTION_NOMATH_COPROC 7

Definition at line 53 of file event.h.

4.46.1.15 #define EXCEPTION_OVERFLOW 4

Definition at line 50 of file event.h.

4.46.1.16 #define EXCEPTION_PAGEFAULT 14

Definition at line 60 of file event.h.

4.46.1.17 #define EXCEPTION_RESERVED 15

Definition at line 61 of file event.h.

4.46.1.18 #define EXCEPTION_SEGMENT_NOT_PRESENT 11

Definition at line 57 of file event.h.

4.46.1.19 #define EXCEPTION_SIMD_FAULT 19

Definition at line 65 of file event.h.

4.46.1.20 #define EXCEPTION_STACKSEG_FAULT 12

Definition at line 58 of file event.h.

4.46.2 Typedef Documentation

4.46.2.1 typedef enum e_idte_type e_idte_type

4.46.3 Enumeration Type Documentation

4.46.3.1 enum e_idte_type

Enumerator

IDTE_TASK

IDTE_INTERRUPT

IDTE_TRAP

Definition at line 26 of file event.h.

```
27 {
28   IDTE_TASK = 5,
29   IDTE_INTERRUPT = 6,
30   IDTE_TRAP = 7
31 } e_idte_type;
```

4.46.4 Function Documentation

4.46.4.1 pok_ret_t pok_event_init ( )

Definition at line 31 of file event.c.

```
32 {
33    pok_idt_init ();
34
35 #if defined (POK_NEEDS_DEBUG) || defined (POK_NEEDS_ERROR_HANDLING)
36    pok_exception_init ();
37 #endif
38
39    pok_syscall_init ();
40
41    return (POK_ERRNO_OK);
42 }
```

4.46.4.2 pok_ret_t pok_exception_init ( )

4.46.4.3 pok_ret_t pok_idt_init ( )

Definition at line 44 of file event.c.

```
45 {
46    sysdesc_t sysdesc;
47
48    /* Clear table */
49    memset(pok_idt, 0, sizeof (idt_entry_t) * IDT_SIZE);
50
51    /* Load IDT */
52    sysdesc.limit = sizeof (pok_idt);
53    sysdesc.base = (uint32_t)pok_idt;
54
55    asm ("lidt %0"
56         :
57         : "m" (sysdesc));
58
59   return (POK_ERRNO_OK);
60 }
```

4.46.4.4 void pok_idt_set_gate (uint16_t *index*, uint16_t *segsel*, uint32_t *offset*, e_idte_type *t*, int *dpl*)

Definition at line 62 of file event.c.

```
67 {
68    pok_idt[index].offset_low   = (offset) & 0xFFFF;
69    pok_idt[index].offset_high  = (offset >> 16) & 0xFFFF;
70    pok_idt[index].segsel       = segsel;
71    pok_idt[index].dpl          = dpl;
72    pok_idt[index].type         = t;
73    pok_idt[index].d            = 1;
74    pok_idt[index].res0         = 0; /* reserved */
75    pok_idt[index].res1         = 0; /* reserved */
76    pok_idt[index].present    = 1;
77 }
```

4.46.4.5 pok_ret_t pok_syscall_init ( )

Init system calls

Definition at line 83 of file syscalls.c.

```
84 {
85    pok_idt_set_gate (POK_SYSCALL_INT_NUMBER,
86                      GDT_CORE_CODE_SEGMENT << 3,
87                      (uint32_t) syscall_gate,
88                      IDTE_INTERRUPT,
89                      3);
90
91    return (POK_ERRNO_OK);
92 }
```

4.47 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/exceptions.c File Reference

4.47.1 Detailed Description

Author

Julian Pidancet

Definition in file exceptions.c.

4.48 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.c File Reference

```
#include <libc.h>
#include <types.h>
#include <errno.h>
#include "gdt.h"
#include "sysdesc.h"
#include "tss.h"
```

Macros

- #define POK_CONFIG_NB_THREADS 0
- #define POK_CONFIG_NB_PARTITIONS 0
- #define GDT_SIZE 256

Functions

- pok_ret_t pok_gdt_init ()
- int pok_tss_init ()

- void tss_set_esp0 (uint32_t esp0)
- void gdt_set_segment (uint16_t index, uint32_t base_address, uint32_t limit, e_gdte_type t, int dpl)
- void gdt_set_system (uint16_t index, uint32_t base_address, uint32_t limit, e_gdte_type t, int dpl)
- void gdt_enable (uint16_t index)
- void gdt_disable (uint16_t index)

Variables

- gdt_entry_t pok_gdt [GDT_SIZE]
- tss_t pok_tss

4.48.1 Macro Definition Documentation

4.48.1.1 #define GDT_SIZE 256

Definition at line 35 of file gdt.c.

4.48.1.2 #define POK_CONFIG_NB_PARTITIONS 0

Definition at line 32 of file gdt.c.

4.48.1.3 #define POK_CONFIG_NB_THREADS 0

Definition at line 28 of file gdt.c.

4.48.2 Function Documentation

4.48.2.1 void gdt_disable (uint16_t *index*)

Definition at line 155 of file gdt.c.

```
156 {
157    pok_gdt[index].present = 0;
158 }
```

4.48.2.2 void gdt_enable (uint16_t *index*)

Definition at line 150 of file gdt.c.

```
151 {
152    pok_gdt[index].present = 1;
153 }
```

4.48.2.3 void gdt_set_segment (uint16_t *index*, uint32_t *base_address*, uint32_t *limit*, e_gdte_type *t*, int *dpl*)

Definition at line 99 of file gdt.c.

```
104 {
105    if (limit > (1 << 20)) /* 4K granularity */
106    {
107       pok_gdt[index].limit_low = (limit >> 12) & 0xFFFF;
108       pok_gdt[index].limit_high = (limit >> 28) & 0xF;
109       pok_gdt[index].granularity = 1;
110    }
```

```
111    else /* 1B granularity */
112    {
113       pok_gdt[index].limit_low = limit & 0xFFFF;
114       pok_gdt[index].limit_high = (limit >> 16) & 0xFF;
115       pok_gdt[index].granularity = 0;
116    }
117
118    pok_gdt[index].base_low = base_address & 0xFFFFFF;
119    pok_gdt[index].base_high = (base_address >> 24) & 0xFF;
120
121    pok_gdt[index].type = t & 0xF;
122    pok_gdt[index].dpl = dpl & 0x3;
123
124    pok_gdt[index].s = 1;               /* Segment is data/code type */
125    pok_gdt[index].present = 1;
126    pok_gdt[index].available = 0;
127    pok_gdt[index].op_size = 1;         /* We work on 32 bits segments */
128 }
```

4.48.2.4 void gdt_set_system (uint16_t *index*, uint32_t *base_address*, uint32_t *limit*, e_gdte_type *t*, int *dpl*)

Definition at line 130 of file gdt.c.

```
135 {
136    pok_gdt[index].limit_low = limit & 0xFFFF;
137    pok_gdt[index].limit_high = (limit >> 16) & 0xFF;
138    pok_gdt[index].base_low = base_address & 0xFFFFFF;
139    pok_gdt[index].base_high = (base_address >> 24) & 0xFF;
140
141    pok_gdt[index].type = t & 0xF;
142    pok_gdt[index].dpl = dpl & 0x3;
143
144    pok_gdt[index].s = 0;               /* Segment is system type */
145    pok_gdt[index].present = 1;
146    pok_gdt[index].available = 0;
147    pok_gdt[index].op_size = 0;
148 }
```

4.48.2.5 pok_ret_t pok_gdt_init ( )

Definition at line 41 of file gdt.c.

```
42 {
43    sysdesc_t sysdesc;
44
45    /* Set null descriptor and clear table */
46    memset(pok_gdt, 0, sizeof (gdt_entry_t) * GDT_SIZE);
47
48    /* Set kernel descriptors */
49    gdt_set_segment(GDT_CORE_CODE_SEGMENT, 0, ~0UL,
      GDTE_CODE, 0);
50    gdt_set_segment(GDT_CORE_DATA_SEGMENT, 0, ~0UL,
      GDTE_DATA, 0);
51
52    /* Load GDT */
53    sysdesc.limit = sizeof (pok_gdt);
54    sysdesc.base = (uint32_t)pok_gdt;
55
56    asm ("lgdt %0"
57          :
58          : "m" (sysdesc));
59
60    /* Reload Segments */
61    asm ("ljmp %0, $1f   \n"
62          "1:            \n"
63          "mov %1, %%ax  \n"
64          "mov %%ax, %%ds        \n"
65          "mov %%ax, %%es        \n"
66          "mov %%ax, %%fs        \n"
67          "mov %%ax, %%gs        \n"
68          "mov %%ax, %%ss        \n"
69          :
70          : "i" (GDT_CORE_CODE_SEGMENT << 3),
71          "i" (GDT_CORE_DATA_SEGMENT << 3)
72          : "eax");
73
74    pok_tss_init();
```

```
75
76    return (POK_ERRNO_OK);
77 }
```

4.48.2.6 int pok_tss_init ( )

Definition at line 79 of file gdt.c.

```
80 {
81    uint16_t sel = GDT_BUILD_SELECTOR(GDT_TSS_SEGMENT, 0, 0);
82
83    memset(&pok_tss, 0, sizeof (tss_t));
84
85    pok_tss.ss0 = GDT_BUILD_SELECTOR(
      GDT_CORE_DATA_SEGMENT, 0, 0);
86
87    gdt_set_system(GDT_TSS_SEGMENT, (uint32_t)&
      pok_tss,
88          sizeof (tss_t), GDTE_TSS, 0);
89
90    asm ("ltr %0" : :"m"(sel));
91    return (POK_ERRNO_OK);
92 }
```

4.48.2.7 void tss_set_esp0 (uint32_t *esp0*)

Definition at line 94 of file gdt.c.

```
95 {
96    pok_tss.esp0 = esp0;
97 }
```

4.48.3 Variable Documentation

4.48.3.1 gdt_entry_t pok_gdt[GDT_SIZE]

Definition at line 37 of file gdt.c.

4.48.3.2 tss_t pok_tss

Definition at line 39 of file gdt.c.

4.49 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.h File Reference

```
#include <types.h>
```

Data Structures

- struct __attribute__

Macros

- #define GDT_CORE_CODE_SEGMENT 1
- #define GDT_CORE_DATA_SEGMENT 2
- #define GDT_TSS_SEGMENT 3

- #define GDT_PARTITION_CODE_SEGMENT(partition_id) (4 + 2 * partition_id)
- #define GDT_PARTITION_DATA_SEGMENT(partition_id) (4 + 2 * partition_id + 1)
- #define GDT_BUILD_SELECTOR(seg, local, rpl) ((seg << 3) | ((local & 0x1) << 2) | (rpl & 0x3))

Typedefs

- typedef enum e_gdte_type e_gdte_type

Enumerations

- enum e_gdte_type { GDTE_CODE = 0xB, GDTE_DATA = 0x3, GDTE_TSS = 0x9 }

Functions

- pok_ret_t pok_gdt_init ()
- int pok_tss_init ()
- void tss_set_esp0 (uint32_t esp0)
- void gdt_set_segment (uint16_t index, uint32_t base_address, uint32_t limit, e_gdte_type t, int dpl)
- void gdt_set_system (uint16_t index, uint32_t base_address, uint32_t limit, e_gdte_type t, int dpl)
- void gdt_enable (uint16_t index)
- void gdt_disable (uint16_t index)

4.49.1 Macro Definition Documentation

4.49.1.1 #define GDT_BUILD_SELECTOR(*seg, local, rpl*) ((seg << 3) | ((local & 0x1) << 2) | (rpl & 0x3))

Definition at line 52 of file gdt.h.

4.49.1.2 #define GDT_CORE_CODE_SEGMENT 1

Definition at line 45 of file gdt.h.

4.49.1.3 #define GDT_CORE_DATA_SEGMENT 2

Definition at line 46 of file gdt.h.

4.49.1.4 #define GDT_PARTITION_CODE_SEGMENT(*partition_id*) (4 + 2 * partition_id)

Definition at line 49 of file gdt.h.

4.49.1.5 #define GDT_PARTITION_DATA_SEGMENT(*partition_id*) (4 + 2 * partition_id + 1)

Definition at line 50 of file gdt.h.

4.49.1.6 #define GDT_TSS_SEGMENT 3

Definition at line 47 of file gdt.h.

4.49.2 Typedef Documentation

4.49.2.1 typedef enum e_gdte_type e_gdte_type

4.49.3 Enumeration Type Documentation

4.49.3.1 enum e_gdte_type

Enumerator

GDTE_CODE

GDTE_DATA

GDTE_TSS

Definition at line 23 of file gdt.h.

```
24 {
25   GDTE_CODE = 0xB,
26   GDTE_DATA = 0x3,
27   GDTE_TSS = 0x9
28 } e_gdte_type;
```

4.49.4 Function Documentation

4.49.4.1 void gdt_disable (uint16_t *index*)

Definition at line 155 of file gdt.c.

```
156 {
157    pok_gdt[index].present = 0;
158 }
```

4.49.4.2 void gdt_enable (uint16_t *index*)

Definition at line 150 of file gdt.c.

```
151 {
152    pok_gdt[index].present = 1;
153 }
```

4.49.4.3 void gdt_set_segment (uint16_t *index*, uint32_t *base_address*, uint32_t *limit*, e_gdte_type *t*, int *dpl*)

Definition at line 99 of file gdt.c.

```
104 {
105    if (limit > (1 << 20)) /* 4K granularity */
106    {
107       pok_gdt[index].limit_low = (limit >> 12) & 0xFFFF;
108       pok_gdt[index].limit_high = (limit >> 28) & 0xF;
109       pok_gdt[index].granularity = 1;
110    }
111    else /* 1B granularity */
112    {
113       pok_gdt[index].limit_low = limit & 0xFFFF;
114       pok_gdt[index].limit_high = (limit >> 16) & 0xFF;
115       pok_gdt[index].granularity = 0;
116    }
117
118    pok_gdt[index].base_low = base_address & 0xFFFFFF;
119    pok_gdt[index].base_high = (base_address >> 24) & 0xFF;
120
121    pok_gdt[index].type = t & 0xF;
122    pok_gdt[index].dpl = dpl & 0x3;
```

```
123
124     pok_gdt[index].s = 1;               /* Segment is data/code type */
125     pok_gdt[index].present = 1;
126     pok_gdt[index].available = 0;
127     pok_gdt[index].op_size = 1;         /* We work on 32 bits segments */
128 }
```

4.49.4.4 void gdt_set_system (uint16_t *index*, uint32_t *base_address*, uint32_t *limit*, e_gdte_type *t*, int *dpl*)

Definition at line 130 of file gdt.c.

```
135 {
136     pok_gdt[index].limit_low = limit & 0xFFFF;
137     pok_gdt[index].limit_high = (limit >> 16) & 0xFF;
138     pok_gdt[index].base_low = base_address & 0xFFFFFF;
139     pok_gdt[index].base_high = (base_address >> 24) & 0xFF;
140
141     pok_gdt[index].type = t & 0xF;
142     pok_gdt[index].dpl = dpl & 0x3;
143
144     pok_gdt[index].s = 0;               /* Segment is system type */
145     pok_gdt[index].present = 1;
146     pok_gdt[index].available = 0;
147     pok_gdt[index].op_size = 0;
148 }
```

4.49.4.5 pok_ret_t pok_gdt_init ( )

Definition at line 41 of file gdt.c.

```
42 {
43     sysdesc_t sysdesc;
44
45     /* Set null descriptor and clear table */
46     memset(pok_gdt, 0, sizeof (gdt_entry_t) * GDT_SIZE);
47
48     /* Set kernel descriptors */
49     gdt_set_segment(GDT_CORE_CODE_SEGMENT, 0, ~0UL,
       GDTE_CODE, 0);
50     gdt_set_segment(GDT_CORE_DATA_SEGMENT, 0, ~0UL,
       GDTE_DATA, 0);
51
52     /* Load GDT */
53     sysdesc.limit = sizeof (pok_gdt);
54     sysdesc.base = (uint32_t)pok_gdt;
55
56     asm ("lgdt %0"
57          :
58          : "m" (sysdesc));
59
60     /* Reload Segments */
61     asm ("ljmp %0, $1f   \n"
62          "1:             \n"
63          "mov %1, %%ax  \n"
64          "mov %%ax, %%ds        \n"
65          "mov %%ax, %%es        \n"
66          "mov %%ax, %%fs        \n"
67          "mov %%ax, %%gs        \n"
68          "mov %%ax, %%ss        \n"
69          :
70          : "i" (GDT_CORE_CODE_SEGMENT << 3),
71          "i" (GDT_CORE_DATA_SEGMENT << 3)
72          : "eax");
73
74     pok_tss_init();
75
76     return (POK_ERRNO_OK);
77 }
```

4.49.4.6 int pok_tss_init ( )

Definition at line 79 of file gdt.c.

```
80 {
81    uint16_t sel = GDT_BUILD_SELECTOR(GDT_TSS_SEGMENT, 0, 0);
82
83    memset(&pok_tss, 0, sizeof (tss_t));
84
85    pok_tss.ss0 = GDT_BUILD_SELECTOR(
      GDT_CORE_DATA_SEGMENT, 0, 0);
86
87    gdt_set_system(GDT_TSS_SEGMENT, (uint32_t)&
      pok_tss,
88          sizeof (tss_t), GDTE_TSS, 0);
89
90    asm ("ltr %0" : :"m"(sel));
91    return (POK_ERRNO_OK);
92 }
```

4.49.4.7 void tss_set_esp0 (uint32_t *esp0*)

Definition at line 94 of file gdt.c.

```
95 {
96    pok_tss.esp0 = esp0;
97 }
```

4.50 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/interrupt.c File Reference

```
#include <arch/x86/interrupt.h>
```

Functions

- void update_tss (interrupt_frame *frame)

4.50.1 Function Documentation

4.50.1.1 void update_tss (interrupt_frame * *frame*)

Definition at line 20 of file interrupt.c.

```
21 {
22   uint32_t* esp0 = (&pok_tss) + 1;
23
24   if ((frame->cs & 0xffff) != 0x8)
25   {
26     *esp0 = (uint32_t)frame + sizeof (interrupt_frame);
27   }
28 }
```

4.51 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/pci.c File Reference

4.52 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/sysdesc.h File Reference

Data Structures

- struct __attribute__

4.53 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/tss.h File Reference

```
#include <types.h>
```

Data Structures

- struct __attribute__

4.54 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/types.h File Reference

Macros

- #define __POK_X86_TYPES_H__

Typedefs

- typedef unsigned short uint8_t
- typedef unsigned short uint16_t
- typedef unsigned int uint32_t
- typedef unsigned long long uint64_t
- typedef short int8_t
- typedef short int16_t
- typedef signed long long int64_t
- typedef unsigned int size_t
- typedef unsigned long int intptr_t

4.54.1 Macro Definition Documentation

4.54.1.1 #define __POK_X86_TYPES_H__

Definition at line 19 of file types.h.

4.54.2 Typedef Documentation

4.54.2.1 typedef short int16_t

Definition at line 27 of file types.h.

4.54.2.2 typedef signed long long int64_t

Definition at line 28 of file types.h.

4.54.2.3 typedef short int8_t

Definition at line 26 of file types.h.

4.54.2.4 typedef unsigned long int intptr_t

Definition at line 31 of file types.h.

4.54.2.5 typedef unsigned int size_t

Definition at line 30 of file types.h.

4.54.2.6 typedef unsigned short uint16_t

Definition at line 22 of file types.h.

4.54.2.7 typedef unsigned int uint32_t

Definition at line 23 of file types.h.

4.54.2.8 typedef unsigned long long uint64_t

Definition at line 24 of file types.h.

4.54.2.9 typedef unsigned short uint8_t

Definition at line 21 of file types.h.

4.55 /home/phipse/gsoc/pok/trunk/kernel/include/arch/sparc/types.h File Reference

Typedefs

- typedef unsigned char uint8_t
- typedef unsigned short uint16_t
- typedef unsigned int uint32_t
- typedef unsigned long long uint64_t
- typedef char int8_t
- typedef short int16_t
- typedef signed long long int64_t
- typedef unsigned int size_t
- typedef unsigned long int intptr_t

4.55.1 Typedef Documentation

4.55.1.1 typedef short int16_t

Definition at line 27 of file types.h.

4.55.1.2 typedef signed long long int64_t

Definition at line 28 of file types.h.

4.55.1.3 typedef char int8_t

Definition at line 26 of file types.h.

4.55.1.4 typedef unsigned long int intptr_t

Definition at line 31 of file types.h.

4.55.1.5 typedef unsigned int size_t

Definition at line 30 of file types.h.

4.55.1.6 typedef unsigned short uint16_t

Definition at line 22 of file types.h.

4.55.1.7 typedef unsigned int uint32_t

Definition at line 23 of file types.h.

4.55.1.8 typedef unsigned long long uint64_t

Definition at line 24 of file types.h.

4.55.1.9 typedef unsigned char uint8_t

Definition at line 21 of file types.h.

4.56 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/types.h File Reference

Macros

- #define __POK_X86_TYPES_H__

Typedefs

- typedef unsigned short uint8_t
- typedef unsigned short uint16_t
- typedef unsigned int uint32_t
- typedef unsigned long long uint64_t
- typedef short int8_t
- typedef short int16_t
- typedef signed long long int64_t
- typedef unsigned int size_t
- typedef unsigned long int intptr_t

4.56.1 Macro Definition Documentation

4.56.1.1 #define __POK_X86_TYPES_H__

Definition at line 19 of file types.h.

4.56.2 Typedef Documentation

4.56.2.1 typedef short int16_t

Definition at line 27 of file types.h.

4.56.2.2 typedef signed long long int64_t

Definition at line 28 of file types.h.

4.56.2.3 typedef short int8_t

Definition at line 26 of file types.h.

4.56.2.4 typedef unsigned long int intptr_t

Definition at line 31 of file types.h.

4.56.2.5 typedef unsigned int size_t

Definition at line 30 of file types.h.

4.56.2.6 typedef unsigned short uint16_t

Definition at line 22 of file types.h.

4.56.2.7 typedef unsigned int uint32_t

Definition at line 23 of file types.h.

4.56.2.8 typedef unsigned long long uint64_t

Definition at line 24 of file types.h.

4.56.2.9 typedef unsigned short uint8_t

Definition at line 21 of file types.h.

4.57 /home/phipse/gsoc/pok/trunk/kernel/include/types.h File Reference

```
#include <arch/x86/types.h>
```

Macros

- #define NULL 0
- #define FALSE 0
- #define TRUE 1
- #define bool_t int
- #define pok_bool_t int

Typedefs

- typedef uint32_t pok_port_size_t
- typedef uint8_t pok_port_direction_t
- typedef uint8_t pok_port_kind_t
- typedef uint8_t pok_queueing_discipline_t
- typedef uint8_t pok_port_id_t
- typedef uint8_t pok_size_t
- typedef uint8_t pok_range_t
- typedef uint8_t pok_buffer_id_t
- typedef uint8_t pok_blackboard_id_t
- typedef uint8_t pok_lockobj_id_t
- typedef uint8_t pok_sem_id_t
- typedef uint8_t pok_event_id_t
- typedef uint8_t pok_partition_id_t
- typedef uint16_t pok_sem_value_t

4.57.1 Macro Definition Documentation

4.57.1.1 #define bool_t int

Definition at line 30 of file types.h.

4.57.1.2 #define FALSE 0

Definition at line 28 of file types.h.

4.57.1.3 #define NULL 0

Definition at line 27 of file types.h.

4.57.1.4 #define pok_bool_t int

Definition at line 31 of file types.h.

4.57.1.5 #define TRUE 1

Definition at line 29 of file types.h.

4.57.2 Typedef Documentation

4.57.2.1 typedef uint8_t pok_blackboard_id_t

Definition at line 41 of file types.h.

4.57.2.2 typedef uint8_t pok_buffer_id_t

Definition at line 40 of file types.h.

4.57.2.3 typedef uint8_t pok_event_id_t

Definition at line 44 of file types.h.

4.57.2.4 typedef uint8_t pok_lockobj_id_t

Definition at line 42 of file types.h.

4.57.2.5 typedef uint8_t pok_partition_id_t

Definition at line 45 of file types.h.

4.57.2.6 typedef uint8_t pok_port_direction_t

Definition at line 34 of file types.h.

4.57.2.7 typedef uint8_t pok_port_id_t

Definition at line 37 of file types.h.

4.57.2.8 typedef uint8_t pok_port_kind_t

Definition at line 35 of file types.h.

4.57.2.9 typedef uint32_t pok_port_size_t

Definition at line 33 of file types.h.

4.57.2.10 typedef uint8_t pok_queueing_discipline_t

Definition at line 36 of file types.h.

4.57.2.11 typedef uint8_t pok_range_t

Definition at line 39 of file types.h.

4.57.2.12 typedef uint8_t pok_sem_id_t

Definition at line 43 of file types.h.

4.57.2.13 typedef uint16_t pok_sem_value_t

Definition at line 46 of file types.h.

4.57.2.14 typedef uint8_t pok_size_t

Definition at line 38 of file types.h.

4.58 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/debug.c File Reference

4.59 /home/phipse/gsoc/pok/trunk/kernel/core/debug.c File Reference

4.60 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pic.c File Reference

```
#include <types.h>
#include <errno.h>
#include <arch/x86/ioports.h>
#include "pic.h"
```

Functions

- int pok_pic_init ()
- int pok_pic_mask (uint8_t irq)
- int pok_pic_unmask (uint8_t irq)
- void pok_pic_eoi (uint8_t irq)

4.60.1 Function Documentation

4.60.1.1 void pok_pic_eoi (uint8_t *irq*)

Definition at line 90 of file pic.c.

```
91 {
92    if (irq >= 8)
93    {
94       outb (PIC_SLAVE_BASE, 0x20);
95    }
96
97    outb (PIC_MASTER_BASE, 0x20);
98 }
```

4.60.1.2 int pok_pic_init ( )

Definition at line 25 of file pic.c.

```
26 {
27    outb (PIC_MASTER_BASE, PIC_MASTER_ICW1);
28    outb (PIC_SLAVE_BASE, PIC_SLAVE_ICW1);
29
30    outb (PIC_MASTER_BASE + 1, PIC_MASTER_ICW2);
31    outb (PIC_SLAVE_BASE + 1, PIC_SLAVE_ICW2);
32
33    outb (PIC_MASTER_BASE + 1, PIC_MASTER_ICW3);
34    outb (PIC_SLAVE_BASE + 1, PIC_SLAVE_ICW3);
35
36    outb (PIC_MASTER_BASE + 1, PIC_MASTER_ICW4);
37    outb (PIC_SLAVE_BASE + 1, PIC_SLAVE_ICW4);
38
39    /* Mask everything */
40    outb (PIC_MASTER_BASE + 1, 0xfb);
41    outb (PIC_SLAVE_BASE + 1, 0xff);
42
43    return (POK_ERRNO_OK);
44 }
```

4.60.1.3 int pok_pic_mask (uint8_t *irq*)

Definition at line 46 of file pic.c.

```
47 {
48    uint8_t mask;
49
50    if (irq > 15)
51    {
52       return (POK_ERRNO_EINVAL);
53    }
54
55    if (irq < 8)
56    {
57       mask = inb (PIC_MASTER_BASE + 1);
58       outb (PIC_MASTER_BASE + 1, mask | (1 << irq));
59    }
60    else
61    {
62       mask = inb (PIC_SLAVE_BASE + 1);
63       outb (PIC_SLAVE_BASE + 1, mask | (1 << (irq - 8)));
64    }
65
66    return (POK_ERRNO_OK);
67 }
```

4.60.1.4 int pok_pic_unmask (uint8_t *irq*)

Definition at line 69 of file pic.c.

```
70 {
71    uint8_t mask;
72
73    if (irq > 15)
74       return (POK_ERRNO_EINVAL);
75
76    if (irq < 8)
77    {
78       mask = inb(PIC_MASTER_BASE + 1);
79       outb(PIC_MASTER_BASE + 1, mask & ~(1 << irq));
80    }
81    else
82    {
83       mask = inb(PIC_SLAVE_BASE + 1);
84       outb(PIC_SLAVE_BASE + 1, mask & ~(1 << (irq - 8)));
85    }
86
87    return (POK_ERRNO_OK);
88 }
```

4.61 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pic.h File Reference

Macros

- #define PIC_MASTER_BASE 0x20
- #define PIC_SLAVE_BASE 0xa0
- #define PIC_MASTER_ICW1 0x11
- #define PIC_MASTER_ICW2 0x20
- #define PIC_MASTER_ICW3 0x04
- #define PIC_MASTER_ICW4 0x01
- #define PIC_SLAVE_ICW1 0x11
- #define PIC_SLAVE_ICW2 0x28
- #define PIC_SLAVE_ICW3 0x02
- #define PIC_SLAVE_ICW4 0x01

Functions

- int pok_pic_init ()
- int pok_pic_mask (uint8_t irq)
- int pok_pic_unmask (uint8_t irq)
- void pok_pic_eoi (uint8_t irq)

4.61.1 Macro Definition Documentation

4.61.1.1 #define PIC_MASTER_BASE 0x20

Definition at line 21 of file pic.h.

4.61.1.2 #define PIC_MASTER_ICW1 0x11

Definition at line 24 of file pic.h.

4.61.1.3 #define PIC_MASTER_ICW2 0x20

Definition at line 25 of file pic.h.

4.61.1.4 #define PIC_MASTER_ICW3 0x04

Definition at line 26 of file pic.h.

4.61.1.5 #define PIC_MASTER_ICW4 0x01

Definition at line 27 of file pic.h.

4.61.1.6 #define PIC_SLAVE_BASE 0xa0

Definition at line 22 of file pic.h.

4.61.1.7 #define PIC_SLAVE_ICW1 0x11

Definition at line 29 of file pic.h.

4.61.1.8 #define PIC_SLAVE_ICW2 0x28

Definition at line 30 of file pic.h.

4.61.1.9 #define PIC_SLAVE_ICW3 0x02

Definition at line 31 of file pic.h.

4.61.1.10 #define PIC_SLAVE_ICW4 0x01

Definition at line 32 of file pic.h.

4.61.2 Function Documentation

4.61.2.1 void pok_pic_eoi (uint8_t *irq*)

Definition at line 90 of file pic.c.

```
91 {
92    if (irq >= 8)
93    {
94       outb (PIC_SLAVE_BASE, 0x20);
95    }
96
97    outb (PIC_MASTER_BASE, 0x20);
98 }
```

4.61.2.2 int pok_pic_init ( )

Definition at line 25 of file pic.c.

```
26 {
27    outb (PIC_MASTER_BASE, PIC_MASTER_ICW1);
28    outb (PIC_SLAVE_BASE, PIC_SLAVE_ICW1);
29
30    outb (PIC_MASTER_BASE + 1, PIC_MASTER_ICW2);
31    outb (PIC_SLAVE_BASE + 1, PIC_SLAVE_ICW2);
32
33    outb (PIC_MASTER_BASE + 1, PIC_MASTER_ICW3);
34    outb (PIC_SLAVE_BASE + 1, PIC_SLAVE_ICW3);
35
36    outb (PIC_MASTER_BASE + 1, PIC_MASTER_ICW4);
37    outb (PIC_SLAVE_BASE + 1, PIC_SLAVE_ICW4);
38
39    /* Mask everything */
40    outb (PIC_MASTER_BASE + 1, 0xfb);
41    outb (PIC_SLAVE_BASE + 1, 0xff);
42
43    return (POK_ERRNO_OK);
44 }
```

4.61.2.3 int pok_pic_mask (uint8_t *irq*)

Definition at line 46 of file pic.c.

```
47 {
48    uint8_t mask;
49
50    if (irq > 15)
51    {
52       return (POK_ERRNO_EINVAL);
53    }
54
55    if (irq < 8)
56    {
57       mask = inb (PIC_MASTER_BASE + 1);
58       outb (PIC_MASTER_BASE + 1, mask | (1 << irq));
59    }
60    else
61    {
62       mask = inb (PIC_SLAVE_BASE + 1);
63       outb (PIC_SLAVE_BASE + 1, mask | (1 << (irq - 8)));
64    }
65
66    return (POK_ERRNO_OK);
67 }
```

4.61.2.4 int pok_pic_unmask (uint8_t *irq*)

Definition at line 69 of file pic.c.

```
70 {
71    uint8_t mask;
72
73    if (irq > 15)
74       return (POK_ERRNO_EINVAL);
75
76    if (irq < 8)
77    {
78       mask = inb(PIC_MASTER_BASE + 1);
79       outb(PIC_MASTER_BASE + 1, mask & ~(1 << irq));
80    }
81    else
82    {
83       mask = inb(PIC_SLAVE_BASE + 1);
84       outb(PIC_SLAVE_BASE + 1, mask & ~(1 << (irq - 8)));
85    }
86
87    return (POK_ERRNO_OK);
88 }
```

4.62 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pit.c File Reference

```
#include <errno.h>
#include <bsp.h>
#include <core/time.h>
#include <core/sched.h>
#include <arch/x86/ioports.h>
#include <arch/x86/interrupt.h>
#include "pic.h"
#include "pit.h"
```

Macros

- #define OSCILLATOR_RATE 1193180 /∗∗ The oscillation rate of x86 clock ∗/
- #define PIT_BASE 0x40
- #define PIT_IRQ 0

Functions

- INTERRUPT_HANDLER (pit_interrupt)
- pok_ret_t pok_x86_qemu_timer_init ()

4.62.1 Macro Definition Documentation

4.62.1.1 #define OSCILLATOR_RATE 1193180 /∗∗ The oscillation rate of x86 clock ∗/

Definition at line 29 of file pit.c.

4.62.1.2 #define PIT_BASE 0x40

Definition at line 30 of file pit.c.

4.62.1.3 #define PIT_IRQ 0

Definition at line 31 of file pit.c.

4.62.2 Function Documentation

4.62.2.1 INTERRUPT_HANDLER (pit_interrupt)

Definition at line 33 of file pit.c.

```
34 {
35    (void) frame;
36    pok_pic_eoi (PIT_IRQ);
37    CLOCK_HANDLER
38 }
```

4.62.2.2 pok_ret_t pok_x86_qemu_timer_init ( )

Definition at line 40 of file pit.c.

```
41 {
42    uint16_t pit_freq;
43
44    pit_freq = POK_TIMER_FREQUENCY;
45
46    outb (PIT_BASE + 3, 0x34); /* Channel0, rate generator, Set LSB then MSB */
47    outb (PIT_BASE, (OSCILLATOR_RATE / pit_freq) & 0xff);
48    outb (PIT_BASE, ((OSCILLATOR_RATE / pit_freq) >> 8) & 0xff);
49
50    pok_bsp_irq_register (PIT_IRQ, pit_interrupt);
51
52    return (POK_ERRNO_OK);
53 }
```

4.63 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pit.h File Reference

Functions

- pok_ret_t pok_x86_qemu_timer_init ()

4.63.1 Function Documentation

4.63.1.1 pok_ret_t pok_x86_qemu_timer_init ( )

Definition at line 40 of file pit.c.

```
41 {
42    uint16_t pit_freq;
43
44    pit_freq = POK_TIMER_FREQUENCY;
45
46    outb (PIT_BASE + 3, 0x34); /* Channel0, rate generator, Set LSB then MSB */
47    outb (PIT_BASE, (OSCILLATOR_RATE / pit_freq) & 0xff);
48    outb (PIT_BASE, ((OSCILLATOR_RATE / pit_freq) >> 8) & 0xff);
49
50    pok_bsp_irq_register (PIT_IRQ, pit_interrupt);
51
52    return (POK_ERRNO_OK);
53 }
```

4.64 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pm.c File Reference

```
#include <errno.h>
#include <arch/x86/multiboot.h>
#include <types.h>
#include "pm.h"
```

Macros

- #define ALIGN_UP(boundary, val) (val + (boundary - 1)) & (~(boundary - 1))

Functions

- int pok_pm_init ()
- uint32_t pok_pm_sbrk (uint32_t increment)

Variables

- void * __pok_begin
- void * __pok_end
- uint32_t pok_multiboot_magic
- uint32_t pok_multiboot_info
- uint32_t pok_x86_pm_heap_start
- uint32_t pok_x86_pm_brk
- uint32_t pok_x86_pm_heap_end

4.64.1 Detailed Description

Author

Julian Pidancet
Julien Delange

Date

2008-2009

Definition in file pm.c.

4.64.2 Macro Definition Documentation

4.64.2.1 #define ALIGN_UP(*boundary, val*) (val + (boundary - 1)) & (~(boundary - 1))

Definition at line 30 of file pm.c.

4.64.3 Function Documentation

4.64.3.1 int pok_pm_init ( )

Definition at line 44 of file pm.c.

```
45 {
46   pok_multiboot_info_t* mbi;
47   uint32_t              free_mem;
48
49   mbi = (pok_multiboot_info_t*) pok_multiboot_info;
50
51 #ifdef POK_NEEDS_DMA
52   free_mem = MEM_16MB;
53 #else
54   free_mem = ALIGN_UP (4096, (uint32_t)(&__pok_end));
55 #endif
56
57   pok_x86_pm_heap_start = pok_x86_pm_brk = free_mem;
58
```

```
59   pok_x86_pm_heap_end = (uint32_t)(mbi->mem_upper * 1024);
60
61   return (POK_ERRNO_OK);
62 }
```

4.64.3.2 uint32_t pok_pm_sbrk (uint32_t *increment*)

Allocation function, very basic, just allocate new memory space each time

Definition at line 68 of file pm.c.

```
69 {
70   uint32_t addr;
71
72   addr = pok_x86_pm_brk;
73
74   pok_x86_pm_brk += increment;
75
76   return (addr);
77 }
```

4.64.4 Variable Documentation

4.64.4.1 void* __pok_begin

4.64.4.2 void* __pok_end

4.64.4.3 uint32_t pok_multiboot_info

4.64.4.4 uint32_t pok_multiboot_magic

4.64.4.5 uint32_t pok_x86_pm_brk

Definition at line 40 of file pm.c.

4.64.4.6 uint32_t pok_x86_pm_heap_end

Definition at line 41 of file pm.c.

4.64.4.7 uint32_t pok_x86_pm_heap_start

Definition at line 39 of file pm.c.

4.65 /home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pm.h File Reference

Macros

- #define MEM_16MB 0x1000000

Functions

- int pok_pm_init ()
- uint32_t pok_pm_sbrk (uint32_t increment)

4.65.1 Macro Definition Documentation

4.65.1.1 #define MEM_16MB 0x1000000

Definition at line 21 of file pm.h.

4.65.2 Function Documentation

4.65.2.1 int pok_pm_init ( )

Definition at line 44 of file pm.c.

```
45 {
46   pok_multiboot_info_t* mbi;
47   uint32_t              free_mem;
48
49   mbi = (pok_multiboot_info_t*) pok_multiboot_info;
50
51 #ifdef POK_NEEDS_DMA
52   free_mem = MEM_16MB;
53 #else
54   free_mem = ALIGN_UP (4096, (uint32_t)(&__pok_end));
55 #endif
56
57   pok_x86_pm_heap_start = pok_x86_pm_brk = free_mem;
58
59   pok_x86_pm_heap_end = (uint32_t)(mbi->mem_upper * 1024);
60
61   return (POK_ERRNO_OK);
62 }
```

4.65.2.2 uint32_t pok_pm_sbrk (uint32_t *increment*)

Allocation function, very basic, just allocate new memory space each time

Definition at line 68 of file pm.c.

```
69 {
70   uint32_t addr;
71
72   addr = pok_x86_pm_brk;
73
74   pok_x86_pm_brk += increment;
75
76   return (addr);
77 }
```

4.66 /home/phipse/gsoc/pok/trunk/kernel/core/boot.c File Reference

Boot function to start the kernel.

```
#include <arch.h>
#include <bsp.h>
#include <core/time.h>
#include <core/thread.h>
#include <core/sched.h>
#include <core/partition.h>
#include <middleware/port.h>
#include <middleware/queue.h>
#include <core/boot.h>
#include <core/instrumentation.h>
```

Functions

- void pok_boot ()

  Boot function that launch everything.

4.66.1 Detailed Description

Boot function to start the kernel.

Author

Julien Delange

Date

2008-2009

Definition in file boot.c.

4.66.2 Function Documentation

4.66.2.1 void pok_boot ( )

Boot function that launch everything.

This function load every service according to system requirements (the POK_NEEDS_∗ maccro). If we don't use partitioning service, we execute a main function. In that case, POK is acting like an executive, not a real kernel

Definition at line 37 of file boot.c.

```
38 {
39    pok_arch_init();
40    pok_bsp_init();
41
42 #if defined (POK_NEEDS_TIME) || defined (POK_NEEDS_SCHED) || defined (POK_NEEDS_THREADS)
43    pok_time_init();
44 #endif
45
46 #ifdef POK_NEEDS_PARTITIONS
47    pok_partition_init ();
48 #endif
49
50 #ifdef POK_NEEDS_THREADS
51    pok_thread_init ();
52 #endif
53
54 #if defined (POK_NEEDS_SCHED) || defined (POK_NEEDS_THREADS)
55    pok_sched_init ();
56 #endif
57
58 #if (defined POK_NEEDS_LOCKOBJ) || defined (POK_NEEDS_PORTS_QUEUEING) || defined (POK_NEEDS_PORTS_SAMPLING)
59    pok_lockobj_init ();
60 #endif
61 #if defined (POK_NEEDS_PORTS_QUEUEING) || defined (POK_NEEDS_PORTS_SAMPLING)
62    pok_port_init ();
63    pok_queue_init ();
64 #endif
65
66 #if defined (POK_NEEDS_DEBUG) || defined (POK_NEEDS_CONSOLE)
67   pok_cons_write ("POK kernel initialized\n", 23);
68 #endif
69
70 #ifdef POK_NEEDS_INSTRUMENTATION
71   uint32_t tmp;
72    printf ("[INSTRUMENTATION][CHEDDAR] <event_table>\n");
73    printf ("[INSTRUMENTATION][CHEDDAR] <processor>\n");
74    printf ("[INSTRUMENTATION][CHEDDAR] <name>pok_kernel</name>\n");
75
76    for (tmp = 0 ; tmp < POK_CONFIG_NB_THREADS ; tmp++)
77    {
78       printf ("[INSTRUMENTATION][CHEDDAR] <task_activation>   0   task %d</task_activation>\n", tmp);
```

```
79    }
80 #endif
81
82   pok_arch_preempt_enable();
83
84 #ifndef POK_NEEDS_PARTITIONS
85
90   main ();
91 #endif
92 }
```

4.67 /home/phipse/gsoc/pok/trunk/kernel/core/error.c File Reference

4.68 /home/phipse/gsoc/pok/trunk/kernel/core/instrumentation.c File Reference

4.69 /home/phipse/gsoc/pok/trunk/kernel/core/kernel.c File Reference

4.70 /home/phipse/gsoc/pok/trunk/kernel/core/loader.c File Reference

4.70.1 Detailed Description

Author

Julian Pidancet
Julien Delange

Date

2008-2009

Contains all needed stuff to load partitions (elf files). This needs the partitioning service (POK_NEEDS_PARTITIONS must be defined) to work.

Definition in file loader.c.

4.71 /home/phipse/gsoc/pok/trunk/kernel/core/lockobj.c File Reference

Provides functionnalities for locking functions (mutexes, semaphores and so on)

4.71.1 Detailed Description

Provides functionnalities for locking functions (mutexes, semaphores and so on)

Author

Julien Delange

This file contains the implementation code for mutexes, conditions and semaphores. This is implemented in the same file since the functionnalities does not differ so much.

Definition in file lockobj.c.

4.72 /home/phipse/gsoc/pok/trunk/kernel/core/partition.c File Reference

This file provides functions for partitioning services.

4.72.1 Detailed Description

This file provides functions for partitioning services.

Author

Julien Delange

The definition of useful structures can be found in partition.h header file. To enable partitioning services, you must set the POK_NEEDS_PARTITIONS maccro.

Definition in file partition.c.

4.73 /home/phipse/gsoc/pok/trunk/kernel/core/sched.c File Reference

4.74 /home/phipse/gsoc/pok/trunk/kernel/core/syscall.c File Reference

```
#include <bsp.h>
#include <types.h>
#include <libc.h>
#include <arch/x86/ioports.h>
#include <arch/x86/pci.h>
#include <errno.h>
#include <core/debug.h>
#include <core/syscall.h>
#include <core/partition.h>
#include <core/thread.h>
#include <core/lockobj.h>
#include <core/time.h>
#include <core/error.h>
#include <middleware/port.h>
```

Functions

- pok_ret_t pok_core_syscall (const pok_syscall_id_t syscall_id, const pok_syscall_args_t *args, const pok_syscall_info_t *infos)

4.74.1 Function Documentation

4.74.1.1 pok_ret_t pok_core_syscall (const pok_syscall_id_t *syscall_id*, const pok_syscall_args_t * *args*, const pok_syscall_info_t * *infos*)

Function that performs the syscall. It is called by the architecture interruption handler.

Parameters

syscall_id	This param correspond to the syscal which should be performed. The list of available syscalls is available in the definition of the pok_syscall_id_t type
args	Arguments of the syscall. It corresponds to data useful to perform the syscall.
infos	Informations about the syscall: which partition/thread initiates the syscall, etc ...

Returns

Returns an error code, which is defined in include/errno.h

Here is the default syscall handler. In this case, the syscall ID was not properly identified and thus, we should return an error. If error management is activated, we raise an error in kernel of partitions, calling the error handler.

Definition at line 40 of file syscall.c.

```
43 {
44    switch (syscall_id)
45    {
46 #if defined (POK_NEEDS_CONSOLE) || defined (POK_NEEDS_DEBUG)
47       case POK_SYSCALL_CONSWRITE:
48          POK_CHECK_PTR_OR_RETURN(infos->partition, args->
      arg1 + infos->base_addr)
49          if (pok_cons_write ((const char*)args->arg1 + infos->base_addr, args->arg2))
50          {
51             return POK_ERRNO_OK;
52          }
53          else
54          {
55             return POK_ERRNO_EINVAL;
56          }
57          break;
58 #endif
59
60 #ifdef POK_NEEDS_PORTS_VIRTUAL
61       case POK_SYSCALL_MIDDLEWARE_VIRTUAL_CREATE:
62          POK_CHECK_PTR_OR_RETURN(infos->partition, args->
      arg1 + infos->base_addr)
63          POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
64          return pok_port_virtual_id ( (char*) (args->arg1 + infos->base_addr), (
      pok_port_id_t*) (args->arg2 + infos->base_addr));
65          break;
66
67       case POK_SYSCALL_MIDDLEWARE_VIRTUAL_NB_DESTINATIONS:
68          POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
69          return pok_port_virtual_nb_destinations ( (pok_port_id_t) (args->arg1), (
      uint32_t*) (args->arg2 + infos->base_addr));
70          break;
71
72       case POK_SYSCALL_MIDDLEWARE_VIRTUAL_DESTINATION:
73          POK_CHECK_PTR_OR_RETURN(infos->partition, ((void*) args->arg3)+infos->
      base_addr)
74          return pok_port_virtual_destination ( (pok_port_id_t) (args->arg1), (
      uint32_t) (args->arg2), (uint32_t*) (args->arg3 + infos->base_addr));
75          break;
76
77       case POK_SYSCALL_MIDDLEWARE_VIRTUAL_GET_GLOBAL:
78          POK_CHECK_PTR_OR_RETURN(infos->partition, (void*) (args->arg2 + infos->
      base_addr))
79          return pok_port_virtual_get_global ((pok_port_id_t) (args->arg1), (
      pok_port_id_t*) (args->arg2 + infos->base_addr));
80          break;
81
82 #endif
83
84 #if defined POK_NEEDS_GETTICK
85       case POK_SYSCALL_GETTICK:
86          POK_CHECK_PTR_OR_RETURN(infos->partition, args->
      arg1 + infos->base_addr)
87          return pok_gettick_by_pointer ((uint64_t*) (args->arg1 + infos->base_addr));
88          break;
89 #endif
90
91       case POK_SYSCALL_THREAD_CREATE:
92          return pok_partition_thread_create  ((uint32_t*)         (args->
      arg1 + infos->base_addr),
93                                              (pok_thread_attr_t*) (args->arg2 + infos->
      base_addr),
94                                              (uint8_t)            infos->
      partition);
95          break;
96
97 #ifdef POK_NEEDS_THREAD_SLEEP
98       case POK_SYSCALL_THREAD_SLEEP:
99          return pok_thread_sleep (args->arg1);
100          break;
101 #endif
102
103 #ifdef POK_NEEDS_THREAD_SLEEP_UNTIL
104       case POK_SYSCALL_THREAD_SLEEP_UNTIL:
105          return pok_thread_sleep_until (args->arg1);
106          break;
```

```
107 #endif
108
109       case POK_SYSCALL_THREAD_PERIOD:
110          return pok_sched_end_period ();
111          break;
112
113 #if defined (POK_NEEDS_THREAD_SUSPEND) || defined (POK_NEEDS_ERROR_HANDLING)
114       case POK_SYSCALL_THREAD_SUSPEND:
115          return pok_thread_suspend ();
116          break;
117 #endif
118
119 #ifdef POK_NEEDS_THREAD_ID
120       case POK_SYSCALL_THREAD_ID:
121          return pok_sched_get_current ((uint32_t*) (args->arg1 + infos->
    base_addr));
122          break;
123 #endif
124    case POK_SYSCALL_THREAD_STATUS:
125               return pok_thread_get_status (args->arg1, (pok_thread_attr_t*) (args->
    arg2 + infos->base_addr));
126          break;
127
128    case POK_SYSCALL_THREAD_DELAYED_START:
129      return pok_thread_delayed_start (args->arg1, args->arg2);
130      break;
131    case POK_SYSCALL_THREAD_SET_PRIORITY:
132           return pok_thread_set_priority (args->arg1, args->arg2);
133           break;
134
135    case POK_SYSCALL_THREAD_RESUME:
136           return pok_thread_resume (args->arg1);
137           break;
138    case POK_SYSCALL_THREAD_SUSPEND_TARGET:
139           return pok_thread_suspend_target (args->arg1);
140           break;
141
142 #ifdef POK_NEEDS_ERROR_HANDLING
143
148       case POK_SYSCALL_THREAD_RESTART:
149          return pok_partition_restart_thread (args->arg1);
150          break;
151
152       case POK_SYSCALL_THREAD_STOP:
153          return pok_partition_stop_thread (args->arg1);
154          break;
155
159       case POK_SYSCALL_THREAD_STOPSELF:
160          pok_sched_stop_self ();
161          return POK_ERRNO_OK;
162          break;
163
164 #endif
165
166 #ifdef POK_NEEDS_PARTITIONS
167       case POK_SYSCALL_PARTITION_SET_MODE:
168          return pok_partition_set_mode_current ((pok_partition_mode_t)args->arg1);
169          break;
170       case POK_SYSCALL_PARTITION_GET_ID:
171         return pok_current_partition_get_id ((uint8_t*)(args->arg1 + infos->
    base_addr));
172          break;
173       case POK_SYSCALL_PARTITION_GET_PERIOD:
174         return pok_current_partition_get_period ((uint64_t*)(args->arg1 + infos->
    base_addr));
175          break;
176       case POK_SYSCALL_PARTITION_GET_DURATION:
177         return pok_current_partition_get_duration ((uint64_t*)(args->
    arg1 + infos->base_addr));
178          break;
179       case POK_SYSCALL_PARTITION_GET_LOCK_LEVEL:
180         return pok_current_partition_get_lock_level ((uint32_t*)(args->
    arg1 + infos->base_addr));
181          break;
182       case POK_SYSCALL_PARTITION_GET_OPERATING_MODE:
183         return pok_current_partition_get_operating_mode ((pok_partition_mode_t*)(args->
    arg1 + infos->base_addr));
184          break;
185       case POK_SYSCALL_PARTITION_GET_START_CONDITION:
186         return pok_current_partition_get_start_condition ((pok_start_condition_t*)(args->
    arg1 + infos->base_addr));
187          break;
188 #endif
189
190 #ifdef POK_NEEDS_ERROR_HANDLING
191       case POK_SYSCALL_ERROR_HANDLER_CREATE:
192          return pok_error_thread_create (args->arg1 , (void*) (args->arg2));
```

```
193         break;
194
195       case POK_SYSCALL_ERROR_RAISE_APPLICATION_ERROR:
196         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg1 + infos->base_addr)
197         pok_error_raise_application_error ((char*) (args->arg1 + infos->base_addr), args->arg2);
198         return POK_ERRNO_OK;
199         break;
200
201       case POK_SYSCALL_ERROR_GET:
202         return pok_error_get ((pok_error_status_t*) (args->arg1 + infos->base_addr));
203         break;
204 #endif
205
206         /* Middleware syscalls */
207 #ifdef POK_NEEDS_PORTS_SAMPLING
208       case POK_SYSCALL_MIDDLEWARE_SAMPLING_CREATE:
209         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg5 + infos->base_addr)
210         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg1 + infos->base_addr)
211         return pok_port_sampling_create  ((char*)(args->arg1 + infos->base_addr),
212                                          (pok_port_size_t) args->arg2,
213                                          (pok_port_direction_t) args->arg3,
214                                          (uint64_t) args->arg4,
215                                          (pok_port_id_t*) (args->arg5 + infos->base_addr));
216         break;
217
218       case POK_SYSCALL_MIDDLEWARE_SAMPLING_WRITE:
219         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
220
221         return pok_port_sampling_write   ((const pok_port_id_t)args->arg1,
222                                          (const void*) ((void*)args->arg2 + infos->base_addr),
223                                          (const uint8_t) args->arg3);
224         break;
225
226       case POK_SYSCALL_MIDDLEWARE_SAMPLING_READ:
227         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
228         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg4 + infos->base_addr)
229         return pok_port_sampling_read ((const pok_port_id_t)args->arg1,
230                                        (void*) args->arg2 + infos->base_addr,
231                                        (pok_port_size_t*) (args->arg3 + infos->base_addr),
232                                        (bool_t*) (args->arg4 + infos->base_addr));
233         break;
234
235       case POK_SYSCALL_MIDDLEWARE_SAMPLING_ID:
236         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg1 + infos->base_addr)
237         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
238         return pok_port_sampling_id   ((char*)(args->arg1 + infos->base_addr),
239                                        (pok_port_id_t*)(args->arg2 + infos->base_addr));
240         break;
241
242 #ifndef POK_GENERATED_CODE
243       case POK_SYSCALL_MIDDLEWARE_SAMPLING_STATUS:
244         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg2+infos->base_addr)
245         return pok_port_sampling_status ((const pok_port_id_t)args->arg1,
246                                          (pok_port_sampling_status_t*) (args->arg2 + infos->base_addr));
247         break;
248 #endif /* POK_GENERATED_CODE */
249 #endif /* POK_NEEDS_PORTS_SAMPLING */
250
251
252 #ifdef POK_NEEDS_PORTS_QUEUEING
253       case POK_SYSCALL_MIDDLEWARE_QUEUEING_CREATE:
254         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg1 + infos->base_addr)
255         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg5 + infos->base_addr)
256         return pok_port_queueing_create ((char*)                          (args->arg1 + infos->base_addr),

257                                          (pok_port_size_t)                args->arg2,
258                                          (pok_port_direction_t)           args->arg3,
259                                          (pok_port_queueing_discipline_t)
    args->arg4,
260                                          (pok_port_id_t*)                 (args->arg5 + infos
    ->base_addr));
261         break;
262
263       case POK_SYSCALL_MIDDLEWARE_QUEUEING_SEND:
264         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
265         return pok_port_queueing_send ((const pok_port_id_t)             args->arg1,
266                                        (const void*)                     ((void*)args->arg2 + infos->
    base_addr),
267                                        (const uint8_t)                   (args->arg3),
268                                        (const uint64_t)                  args->arg4);
269         break;
270
271       case POK_SYSCALL_MIDDLEWARE_QUEUEING_RECEIVE:
```

```
272         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg4 + infos->base_addr)
273         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg5 + infos->base_addr)
274         return pok_port_queueing_receive ((const pok_port_id_t)   args->arg1,
275                                           (uint64_t)              args->arg2,
276                                           (pok_port_size_t)       args->arg3,
277                                           (void*)                 ((void*)args->arg4 + infos->base_addr),
278                                           (pok_port_size_t*)      (args->arg5 + infos->
    base_addr));
279         break;
280
281      case POK_SYSCALL_MIDDLEWARE_QUEUEING_ID:
282         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg1 + infos->base_addr)
283         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
284         return pok_port_queueing_id ((char*)          (args->arg1 + infos->base_addr),
285                                      (pok_port_id_t*)  (args->arg2 + infos->base_addr));
286         break;
287
288 #ifndef POK_GENERATED_CODE
289      case POK_SYSCALL_MIDDLEWARE_QUEUEING_STATUS:
290         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg2 + infos->base_addr)
291         return pok_port_queueing_status     ((const pok_port_id_t)           args->arg1,
292                                             (pok_port_queueing_status_t*)    (args->arg2 + infos->
    base_addr));
293         break;
294 #endif
295 #endif /* POK_NEEDS_PORTS_QUEUEING */
296
297 #ifdef POK_NEEDS_LOCKOBJECTS
298      case POK_SYSCALL_LOCKOBJ_CREATE:
299         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg2+infos->base_addr)
300         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg1+infos->base_addr)
301         return pok_lockobj_partition_create    ((
    pok_lockobj_id_t*)    (args->arg1 + infos->base_addr),
302                                                (pok_lockobj_attr_t*)   (args->arg2 +
    infos->base_addr));
303         break;
304
305      case POK_SYSCALL_LOCKOBJ_OPERATION:
306         if (args->arg2 == NULL)
307         {
308            return pok_lockobj_partition_wrapper ((const
    uint8_t) args->arg1, NULL);
309         }
310         else
311         {
312            POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg2 + infos->base_addr)
313            return pok_lockobj_partition_wrapper   ((const
    uint8_t) args->arg1,
314                                                   (pok_lockobj_lockattr_t*) (args->
    arg2 + infos->base_addr));
315         }
316         break;
317 #endif /* POK_NEEDS_LOCKOBJECTS */
318
319 #ifdef POK_NEEDS_IO
320      case POK_SYSCALL_INB:
321         if ((args->arg1 < pok_partitions[infos->partition].io_min) ||
322             (args->arg1 > pok_partitions[infos->partition].io_max))
323         {
324            return -POK_ERRNO_EPERM;
325         }
326         else
327         {
328            return inb((unsigned short) args->arg1);
329         }
330         break;
331
332      case POK_SYSCALL_OUTB:
333         if ((args->arg1 < pok_partitions[infos->partition].io_min) ||
334             (args->arg1 > pok_partitions[infos->partition].io_max))
335         {
336            return -POK_ERRNO_EPERM;
337         }
338         else
339         {
340            outb((unsigned short) args->arg1, (unsigned char) args->arg2);
341            return POK_ERRNO_OK;
342         }
343       break;
344 #endif /* POK_NEEDS_IO */
345
346 #ifdef POK_NEEDS_PCI
347     case POK_SYSCALL_PCI_REGISTER:
348       POK_CHECK_PTR_OR_RETURN(infos->partition, args->
```

```
    arg1 + infos->base_addr)
349       return pci_register((void*)args->arg1 + infos->base_addr, infos->partition);
350       break;
351 #endif /* POK_NEEDS_PCI */
352
353
360     default:
361 #ifdef POK_NEEDS_ERROR_HANDLING
362         pok_error_declare (POK_ERROR_KIND_ILLEGAL_REQUEST);
363         pok_sched_activate_error_thread ();
364 #else
365         #ifdef POK_NEEDS_DEBUG
366            printf ("Tried to use syscall %d\n", syscall_id);
367         #endif
368         POK_FATAL ("Unknown syscall");
369 #endif
370         break;
371   }
372
373   return POK_ERRNO_EINVAL;
374 }
```

4.75 /home/phipse/gsoc/pok/trunk/kernel/core/time.c File Reference

4.75.1 Detailed Description

Author

François Goudal
Julien Delange

Date

2008-2009

Definition in file time.c.

4.76 /home/phipse/gsoc/pok/trunk/kernel/include/arch.h File Reference

Generic interface to handle architectures.

```
#include <types.h>
#include <errno.h>
```

Functions

- pok_ret_t pok_arch_init ()
- pok_ret_t pok_arch_preempt_disable ()
- pok_ret_t pok_arch_preempt_enable ()
- pok_ret_t pok_arch_idle ()
- pok_ret_t pok_arch_event_register (uint8_t vector, void(∗handler)(void))
- uint32_t pok_context_create (uint32_t thread_id, uint32_t stack_size, uint32_t entry)
- void pok_context_switch (uint32_t ∗old_sp, uint32_t new_sp)
- void pok_context_reset (uint32_t stack_size, uint32_t stack_addr)
- pok_ret_t pok_create_space (uint8_t partition_id, uint32_t addr, uint32_t size)
- uint32_t pok_space_base_vaddr (uint32_t addr)
- void pok_dispatch_space (uint8_t partition_id, uint32_t user_pc, uint32_t user_sp, uint32_t kernel_sp, uint32_t arg1, uint32_t arg2)
- uint32_t pok_space_context_create (uint8_t partition_id, uint32_t entry_rel, uint32_t stack_rel, uint32_t arg1, uint32_t arg2)

- void pok_space_context_restart (uint32_t sp, uint32_t entry, uint32_t user_stack)
- pok_ret_t pok_space_switch (uint8_t old_partition_id, uint8_t new_partition_id)
- uint32_t pok_thread_stack_addr (const uint8_t partition_id, const uint32_t local_thread_id)

4.76.1 Detailed Description

Generic interface to handle architectures.

Author

Julian Pidancet
Julien Delange

Date

2008-2009

Definition in file arch.h.

4.76.2 Function Documentation

4.76.2.1 pok_ret_t pok_arch_event_register (uint8_t *vector,* void(*)(void) *handler*)

Register an event (for example, an interruption)

Attach the handler to the given trap number (vector).

See Also

pok_sparc_isr

Definition at line 83 of file arch.c.

```
84 {
85   (void) vector;
86   (void) handler;
87
88   return (POK_ERRNO_OK);
89 }
```

4.76.2.2 pok_ret_t pok_arch_idle ( )

Function that do nothing. Useful for the idle task for example.

Definition at line 74 of file arch.c.

```
75 {
76    while (1)
77    {
78    }
79
80    return (POK_ERRNO_OK);
81 }
```

4.76.2.3 pok_ret_t pok_arch_init ( )

Function that initializes architecture concerns.

Initialize all SPARC managers (traps, syscalls, space).

Definition at line 43 of file arch.c.

```
44 {
45   set_msr (MSR_IP);
46 #if POK_NEEDS_PARTITIONS
47   pok_arch_space_init();
48 #endif
49
50   return (POK_ERRNO_OK);
51 }
```

4.76.2.4 pok_ret_t pok_arch_preempt_disable ( )

Disable interruptions

Definition at line 53 of file arch.c.

```
54 {
55   unsigned int msr;
56
57   msr = get_msr();
58   msr &= ~MSR_EE;
59   set_msr(msr);
60   return (POK_ERRNO_OK);
61 }
```

4.76.2.5 pok_ret_t pok_arch_preempt_enable ( )

Enable interruptions

Definition at line 63 of file arch.c.

```
64 {
65   unsigned int msr;
66
67   msr = get_msr();
68   msr |= MSR_EE;
69   set_msr(msr);
70
71   return (POK_ERRNO_OK);
72 }
```

4.76.2.6 uint32_t pok_context_create (uint32_t *thread_id*, uint32_t *stack_size*, uint32_t *entry*)

4.76.2.7 void pok_context_reset (uint32_t *stack_size*, uint32_t *stack_addr*)

4.76.2.8 void pok_context_switch (uint32_t * *old_sp*, uint32_t *new_sp*)

4.76.2.9 pok_ret_t pok_create_space (uint8_t *partition_id*, uint32_t *addr*, uint32_t *size*)

Set ptd and pte for the given partition.

Definition at line 42 of file space.c.

```
45 {
46 #ifdef POK_NEEDS_DEBUG
47   printf ("pok_create_space: %d: %x %x\n", partition_id, addr, size);
48 #endif
49   spaces[partition_id].phys_base = addr;
```

```
50   spaces[partition_id].size = size;
51
52   return (POK_ERRNO_OK);
53 }
```

4.76.2.10 void pok_dispatch_space (uint8_t *partition_id*, uint32_t *user_pc*, uint32_t *user_sp*, uint32_t *kernel_sp*, uint32_t *arg1*, uint32_t *arg2*)

Definition at line 114 of file space.c.

```
120 {
121    interrupt_frame   ctx;
122    uint32_t          code_sel;
123    uint32_t          data_sel;
124    uint32_t          sp;
125
126    code_sel = GDT_BUILD_SELECTOR (GDT_PARTITION_CODE_SEGMENT (
      partition_id), 0, 3);
127    data_sel = GDT_BUILD_SELECTOR (GDT_PARTITION_DATA_SEGMENT (
      partition_id), 0, 3);
128
129    sp = (uint32_t) &ctx;
130
131    memset (&ctx, 0, sizeof (interrupt_frame));
132
133    pok_arch_preempt_disable ();
134
135    ctx.es = ctx.ds = ctx.ss = data_sel;
136
137    ctx.__esp    = (uint32_t) (&ctx.error); /* for pusha */
138    ctx.eip      = user_pc;
139    ctx.eax      = arg1;
140    ctx.ebx      = arg2;
141    ctx.cs       = code_sel;
142    ctx.eflags   = 1 << 9;
143    ctx.esp      = user_sp;
144
145    tss_set_esp0 (kernel_sp);
146
147    asm ("mov %0, %%esp          \n"
148         "pop %%es               \n"
149         "pop %%ds               \n"
150         "popa                   \n"
151         "addl $4, %%esp         \n"
152         "iret                   \n"
153         :
154         : "m" (sp)
155        );
156 }
```

4.76.2.11 uint32_t pok_space_base_vaddr (uint32_t *addr*)

Returns

partition virtual base adress.

See Also

SPARC_PARTITION_BASE_VADDR

Definition at line 64 of file space.c.

```
65 {
66    (void) addr;
67    return (0);
68 }
```

4.76.2.12 uint32_t pok_space_context_create (uint8_t *id*, uint32_t *entry_rel*, uint32_t *stack_rel*, uint32_t *arg1*, uint32_t *arg2*)

Create a new context in the given space

Initilize thread stack.

Definition at line 72 of file space.c.

```
77 {
78   context_t* ctx;
79   volatile_context_t* vctx;
80   char*      stack_addr;
81   (void) partition_id;
82
83   stack_addr = pok_bsp_mem_alloc (KERNEL_STACK_SIZE);
84
85   vctx = (volatile_context_t *)
86     (stack_addr + KERNEL_STACK_SIZE - sizeof (
      volatile_context_t));
87   ctx = (context_t *)((char *)vctx - sizeof (context_t) + 8);
88
89   memset (ctx, 0, sizeof (*ctx));
90   memset (vctx, 0, sizeof (*vctx));
91
92   vctx->r3     = arg1;
93   vctx->r4     = arg2;
94   vctx->sp     = stack_rel - 12;
95   vctx->srr0   = entry_rel;
96   vctx->srr1   = MSR_EE | MSR_IP | MSR_DR | MSR_IR |
      MSR_PR;
97   ctx->lr      = (uint32_t) pok_arch_rfi;
98
99   ctx->sp      = (uint32_t) &vctx->sp;
100
101 #ifdef POK_NEEDS_DEBUG
102   printf ("space_context_create %d: entry=%x stack=%x arg1=%x arg2=%x ksp=%x\n",
103           partition_id, entry_rel, stack_rel, arg1, arg2, &vctx->sp);
104 #endif
105
106   return (uint32_t)ctx;
107 }
```

4.76.2.13 void pok_space_context_restart (uint32_t *sp*, uint32_t *entry*, uint32_t *user_stack*)

4.76.2.14 pok_ret_t pok_space_switch (uint8_t *old_partition_id*, uint8_t *new_partition_id*)

Switch from one space to another

Switch adress space in MMU (context register).

Definition at line 55 of file space.c.

```
57 {
58   (void) old_partition_id;
59   /* printf ("space_switch %u -> %u\n", old_partition_id, new_partition_id); */
60   asm volatile ("mtsr %0,%1" : : "r"(0), "r"(PPC_SR_KP | new_partition_id));
61   return (POK_ERRNO_OK);
62 }
```

4.76.2.15 uint32_t pok_thread_stack_addr (const uint8_t *partition_id*, const uint32_t *local_thread_id*)

Returns the stack address for a the thread number N in a partition.

- partition_id indicates the partition that contains the thread.
- local_thread_id the thread-id of the thread inside the partition.

Returns

the stack address of the thread.

Compute the stack adress for the given thread.

Definition at line 92 of file arch.c.

```
94 {
95    return pok_partitions[partition_id].size - 16 - (local_thread_id * POK_USER_STACK_SIZE);
96 }
```

4.77 /home/phipse/gsoc/pok/trunk/kernel/include/arch/ppc/spinlock.h File Reference

Macros

- #define SPIN_UNLOCK(_spin_) (_spin_) = 0
- #define SPIN_LOCK(_spin_)

Typedefs

- typedef unsigned int pok_spinlock_t

4.77.1 Macro Definition Documentation

4.77.1.1 #define SPIN_LOCK(*_spin_*)

Value:

```
do {                                                    \
    unsigned int val;                                     \
    asm volatile ("\n"                                    \
                  "1:\n\t"                                \
                  "lwarx    %0,0,%1       \n\t"           \
                  "cmpwi    %0,0          \n\t"           \
                  "bne      1b            \n\t"           \
                  "stwcx.   %2,0,%1       \n\t"           \
                  "bne      1b            \n\t"           \
                  : "=&r"(val) : "r" (&_spin_), "r"(1));    \
  } while (0)
```

Definition at line 26 of file spinlock.h.

4.77.1.2 #define SPIN_UNLOCK(*_spin_*) (_spin_) = 0

Definition at line 23 of file spinlock.h.

4.77.2 Typedef Documentation

4.77.2.1 typedef unsigned int pok_spinlock_t

Definition at line 21 of file spinlock.h.

4.78 /home/phipse/gsoc/pok/trunk/kernel/include/arch/sparc/spinlock.h File Reference

Macros

- #define SPIN_UNLOCK(_spin_) (_spin_) = 0
- #define SPIN_LOCK(_spin_)

Typedefs

- typedef unsigned int pok_spinlock_t

4.78.1 Macro Definition Documentation

4.78.1.1 #define SPIN_LOCK(*_spin_*)

Value:

```
do {                                            \
    asm volatile ("1:                 \n"         \
                  "ldstub [%0], %1 \n"          \
                  "tst %1             \n"         \
                  "bnz 1b             \n"         \
                  : /* no output */             \
                  : "r" (&(_spin_)), "r"(1));   \
  } while (0)
```

Definition at line 30 of file spinlock.h.

4.78.1.2 #define SPIN_UNLOCK(*_spin_*) (_spin_) = 0

Definition at line 27 of file spinlock.h.

4.78.2 Typedef Documentation

4.78.2.1 typedef unsigned int pok_spinlock_t

Definition at line 25 of file spinlock.h.

4.79 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/spinlock.h File Reference

Macros

- #define SPIN_UNLOCK(_spin_)
- #define SPIN_LOCK(_spin_)

Typedefs

- typedef unsigned char pok_spinlock_t

4.79.1 Macro Definition Documentation

4.79.1.1 #define SPIN_LOCK(_spin_)

Value:

```
asm volatile ("mov $1, %%al           \n\t"                   \
                "1:                     \n\t"                   \
                "lock xchg %0, %%al     \n\t"                   \
                "test %%al, %%al        \n\t"                   \
                "jnz 1b                 \n\t"                   \
                :                                               \
                : "m" (_spin_)                                  \
                : "%al")
```

Definition at line 28 of file spinlock.h.

4.79.1.2 #define SPIN_UNLOCK(_spin_)

Value:

```
{                                                               \
  (_spin_) = 0;                                                 \
}
```

Definition at line 23 of file spinlock.h.

4.79.2 Typedef Documentation

4.79.2.1 typedef unsigned char pok_spinlock_t

Definition at line 21 of file spinlock.h.

4.80 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/interrupt.h File Reference

```
#include <types.h>
```

Data Structures

- struct interrupt_frame

Macros

- #define INTERRUPT_HANDLER(name)
- #define INTERRUPT_HANDLER_errorcode(name)
- #define INTERRUPT_HANDLER_syscall(name)

Functions

- void update_tss (interrupt_frame *frame)

Variables

- uint32_t pok_tss

4.80.1 Macro Definition Documentation

4.80.1.1 #define INTERRUPT_HANDLER(*name*)

Value:

```
void name (void);                                                       \
void name##_handler(interrupt_frame* frame);                             \
  asm (                                                                 \
       ".global "#name "                  \n"                          \
       "\t.type "#name",@function         \n"                          \
       #name":                            \n"                          \
       "cli                         \n"                          \
       "subl $4, %esp                     \n"                          \
       "pusha                             \n"                          \
       "push %ds                          \n"                          \
       "push %es                          \n"                          \
       "push %esp                         \n"                          \
       "mov $0x10, %ax                    \n"                          \
       "mov %ax, %ds                      \n"                          \
       "mov %ax, %es                      \n"                          \
       "call " #name"_handler             \n"                          \
       "call update_tss                   \n"                          \
       "addl $4, %esp                     \n"                          \
       "pop %es                           \n"                          \
       "pop %ds                           \n"                          \
       "popa                              \n"                          \
       "addl $4, %esp                     \n"                          \
       "sti                         \n"                          \
       "iret                              \n"                          \
       );                                                               \
void name##_handler(interrupt_frame* frame)
```

Definition at line 53 of file interrupt.h.

4.80.1.2 #define INTERRUPT_HANDLER_errorcode(*name*)

Value:

```
void name (void);                                                       \
void name##_handler(interrupt_frame* frame);                             \
  asm (                                                                 \
       ".global "#name "                  \n"                          \
       "\t.type "#name",@function         \n"                          \
       #name":                            \n"                          \
       "cli                         \n"                          \
       "pusha                             \n"                          \
       "push %ds                          \n"                          \
       "push %es                          \n"                          \
       "push %esp                         \n"                          \
       "mov $0x10, %ax                    \n"                          \
       "mov %ax, %ds                      \n"                          \
       "mov %ax, %es                      \n"                          \
       "call " #name"_handler             \n"                          \
       "call update_tss                   \n"                          \
       "addl $4, %esp                     \n"                          \
       "pop %es                           \n"                          \
       "pop %ds                           \n"                          \
       "popa                              \n"                          \
       "addl $4, %esp                     \n"                          \
       "sti                         \n"                          \
       "iret                              \n"                          \
       );                                                               \
void name##_handler(interrupt_frame* frame)
```

Definition at line 81 of file interrupt.h.

4.80.1.3 #define INTERRUPT_HANDLER_syscall(*name*)

Value:

```
int name (void);                                                        \
void name##_handler(interrupt_frame* frame);                             \
```

```
  asm (                                                                 \
      ".global "#name "                 \n"                             \
      "\t.type "#name",@function        \n"                             \
      #name":                           \n"                             \
      "cli                      \n"                                     \
      "subl $4, %esp                    \n"                             \
      "pusha                            \n"                             \
      "push %ds                         \n"                             \
      "push %es                         \n"                             \
      "push %esp                        \n"                             \
      "mov $0x10, %ax                   \n"                             \
      "mov %ax, %ds                     \n"                             \
      "mov %ax, %es                     \n"                             \
      "call " #name"_handler            \n"                             \
      "movl %eax, 40(%esp)          \n" /* return value */  \
      "call update_tss                  \n"                             \
      "addl $4, %esp                    \n"                             \
      "pop %es                          \n"                             \
      "pop %ds                          \n"                             \
      "popa                             \n"                             \
      "addl $4, %esp                    \n"                             \
      "sti                      \n"                                     \
      "iret                             \n"                             \
      );                                                                \
void name##_handler(interrupt_frame* frame)
```

Definition at line 108 of file interrupt.h.

4.80.2 Function Documentation

4.80.2.1 void update_tss (interrupt_frame * *frame*)

Definition at line 20 of file interrupt.c.

```
21 {
22   uint32_t* esp0 = (&pok_tss) + 1;
23
24   if ((frame->cs & 0xffff) != 0x8)
25   {
26     *esp0 = (uint32_t)frame + sizeof (interrupt_frame);
27   }
28 }
```

4.80.3 Variable Documentation

4.80.3.1 uint32_t pok_tss

Definition at line 39 of file gdt.c.

4.81 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.h File Reference

Data Structures

- struct pok_multiboot_header_t
- struct pok_aout_symbol_table_t
- struct pok_elf_section_header_table_t
- struct pok_multiboot_info_t
- struct pok_module_t
- struct pok_memory_map_t

Macros

- #define MULTIBOOT_BOOTLOADER_MAGIC 0x2BADB002

- #define MULTIBOOT_HEADER_MAGIC 0x1BADB002
- #define MULTIBOOT_HEADER_FLAGS 0x00010003
- #define MULTIBOOT_BOOTLOADER_MAGIC 0x2BADB002
- #define MULTIBOOT_STACK_SIZE 0x4000
- #define MULTIBOOT_CMDLINE 4
- #define MULTIBOOT_MODS 8
- #define EXT_C(sym) sym

4.81.1 Detailed Description

Author

Julien Pidancet

Date

2008-2009

Definition in file multiboot.h.

4.81.2 Macro Definition Documentation

4.81.2.1 #define EXT_C(*sym*) sym

C symbol format. HAVE_ASM_USCORE is defined by configure.

Definition at line 57 of file multiboot.h.

4.81.2.2 #define MULTIBOOT_BOOTLOADER_MAGIC 0x2BADB002

The magic number passed by a Multiboot-compliant boot loader.

Definition at line 41 of file multiboot.h.

4.81.2.3 #define MULTIBOOT_BOOTLOADER_MAGIC 0x2BADB002

The magic number passed by a Multiboot-compliant boot loader.

Definition at line 41 of file multiboot.h.

4.81.2.4 #define MULTIBOOT_CMDLINE 4

Definition at line 48 of file multiboot.h.

4.81.2.5 #define MULTIBOOT_HEADER_FLAGS 0x00010003

The flags for the Multiboot header.

Definition at line 36 of file multiboot.h.

4.81.2.6 #define MULTIBOOT_HEADER_MAGIC 0x1BADB002

The magic number for the Multiboot header.

Definition at line 31 of file multiboot.h.

4.81.2.7 #define MULTIBOOT_MODS 8

Definition at line 49 of file multiboot.h.

4.81.2.8 #define MULTIBOOT_STACK_SIZE 0x4000

The size of our stack (16KB).

Definition at line 46 of file multiboot.h.

4.82 /home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/pci.h File Reference

4.83 /home/phipse/gsoc/pok/trunk/kernel/include/bsp.h File Reference

Interfaces that BSP must provide.

```
#include <types.h>
#include <errno.h>
```

Functions

- pok_ret_t pok_bsp_init ()
- pok_ret_t pok_bsp_irq_acknowledge (uint8_t irq)
- pok_ret_t pok_bsp_irq_register (uint8_t irq, void(*handler)(void))
- void * pok_bsp_mem_alloc (size_t size)
- pok_ret_t pok_bsp_time_init ()
- bool_t pok_cons_write (const char *s, size_t length)

4.83.1 Detailed Description

Interfaces that BSP must provide.

Author

Julian Pidancet

Date

2008-2009

Definition in file bsp.h.

4.83.2 Function Documentation

4.83.2.1 pok_ret_t pok_bsp_init ( )

Definition at line 22 of file bsp.c.

```
23 {
24    pok_cons_init ();
25
26    return (POK_ERRNO_OK);
27 }
```

4.83.2.2 pok_ret_t pok_bsp_irq_acknowledge (uint8_t *irq*)

Definition at line 35 of file bsp.c.

```
36 {
37    pok_pic_eoi (irq);
38
39    return (POK_ERRNO_OK);
40 }
```

4.83.2.3 pok_ret_t pok_bsp_irq_register (uint8_t *irq*, void(∗)(void) *handler*)

Definition at line 42 of file bsp.c.

```
44 {
45    pok_pic_unmask (irq);
46
47    pok_arch_event_register (32 + irq, handler);
48
49    return (POK_ERRNO_OK);
50 }
```

4.83.2.4 void∗ pok_bsp_mem_alloc (size_t *size*)

Used for partition allocation. For SPARC support, all partitions are aligned on page size and all partition sizes have to be less than page size.

See Also

SPARC_PAGE_SIZE

Allocate data. At this time, the pok_pm_sbrk function only increment size each time we allocate memory and was not designed to free previously allocated memory.

Definition at line 34 of file bsp.c.

```
35 {
36   char *res;
37
38   res = (char *)(((unsigned int)heap_end + 4095) & ~4095);
39   heap_end = res + sz;
40   return res;
41 }
```

4.83.2.5 pok_ret_t pok_bsp_time_init ( )

Initialize the timer, register the ISR and unmask the interrupt.

See Also

unmask_irq(irq_nbr)

Init time. *freq* is the frequency of the oscillator.

Definition at line 87 of file timer.c.

```
88 {
89   time_inter = (BUS_FREQ * FREQ_DIV) / POK_TIMER_FREQUENCY;
90   time_last = get_ppc_tb ();
91   pok_arch_set_decr();
92
93   return (POK_ERRNO_OK);
94 }
```

4.83.2.6 bool_t pok_cons_write (const char * *s*, size_t *length*)

4.84 /home/phipse/gsoc/pok/trunk/kernel/include/core/boot.h File Reference

Functions

- void pok_boot ()

  Boot function that launch everything.

4.84.1 Detailed Description

Author

Julien Delange

Date

2008-2009

Definition in file boot.h.

4.84.2 Function Documentation

4.84.2.1 void pok_boot ( )

Boot function that launch everything.

This function load every service according to system requirements (the POK_NEEDS_* maccro). If we don't use partitioning service, we execute a main function. In that case, POK is acting like an executive, not a real kernel

Definition at line 37 of file boot.c.

```
38 {
39    pok_arch_init();
40    pok_bsp_init();
41
42 #if defined (POK_NEEDS_TIME) || defined (POK_NEEDS_SCHED) || defined (POK_NEEDS_THREADS)
43    pok_time_init();
44 #endif
45
46 #ifdef POK_NEEDS_PARTITIONS
47    pok_partition_init ();
48 #endif
49
50 #ifdef POK_NEEDS_THREADS
51    pok_thread_init ();
52 #endif
53
54 #if defined (POK_NEEDS_SCHED) || defined (POK_NEEDS_THREADS)
55    pok_sched_init ();
56 #endif
57
58 #if (defined POK_NEEDS_LOCKOBJ) || defined (POK_NEEDS_PORTS_QUEUEING) || defined (POK_NEEDS_PORTS_SAMPLING)
59    pok_lockobj_init ();
60 #endif
61 #if defined (POK_NEEDS_PORTS_QUEUEING) || defined (POK_NEEDS_PORTS_SAMPLING)
62    pok_port_init ();
63    pok_queue_init ();
64 #endif
65
66 #if defined (POK_NEEDS_DEBUG) || defined (POK_NEEDS_CONSOLE)
67   pok_cons_write ("POK kernel initialized\n", 23);
68 #endif
69
70 #ifdef POK_NEEDS_INSTRUMENTATION
71   uint32_t tmp;
72    printf ("[INSTRUMENTATION][CHEDDAR] <event_table>\n");
73    printf ("[INSTRUMENTATION][CHEDDAR] <processor>\n");
```

```
74    printf ("[INSTRUMENTATION][CHEDDAR] <name>pok_kernel</name>\n");
75
76    for (tmp = 0 ; tmp < POK_CONFIG_NB_THREADS ; tmp++)
77    {
78       printf ("[INSTRUMENTATION][CHEDDAR] <task_activation>   0   task %d</task_activation>\n", tmp);
79    }
80 #endif
81
82   pok_arch_preempt_enable();
83
84 #ifndef POK_NEEDS_PARTITIONS
85
90   main ();
91 #endif
92 }
```

4.85 /home/phipse/gsoc/pok/trunk/kernel/include/core/cpio.h File Reference

Data Structures

- struct cpio_bin_header
- struct cpio_file

Enumerations

- enum cpio_format {
  CPIO_BIN_FMT, CPIO_ODC_FMT, CPIO_NEWC_FMT, CPIO_CRC_FMT,
  CPIO_TAR_FMT, CPIO_USTAR_FMT, CPIO_HPBIN_FMT, CPIO_HPODC_FMT }

Functions

- int cpio_open (struct cpio_file *cpio, void *file)
- char * cpio_get_filename (struct cpio_file *cpio)
- int cpio_next_file (struct cpio_file *cpio)
- void * cpio_get_fileaddr (struct cpio_file *cpio)

4.85.1 Enumeration Type Documentation

4.85.1.1 enum cpio_format

Enumerator

CPIO_BIN_FMT

CPIO_ODC_FMT

CPIO_NEWC_FMT

CPIO_CRC_FMT

CPIO_TAR_FMT

CPIO_USTAR_FMT

CPIO_HPBIN_FMT

CPIO_HPODC_FMT

Definition at line 21 of file cpio.h.

```
22 {
23    CPIO_BIN_FMT,
24    CPIO_ODC_FMT,
25    CPIO_NEWC_FMT,
26    CPIO_CRC_FMT,
```

```
27     CPIO_TAR_FMT,
28     CPIO_USTAR_FMT,
29     CPIO_HPBIN_FMT,
30     CPIO_HPODC_FMT
31 };
```

4.85.2 Function Documentation

4.85.2.1 void* cpio_get_fileaddr (struct cpio_file * *cpio*)

4.85.2.2 char* cpio_get_filename (struct cpio_file * *cpio*)

4.85.2.3 int cpio_next_file (struct cpio_file * *cpio*)

4.85.2.4 int cpio_open (struct cpio_file * *cpio*, void * *file*)

4.86 /home/phipse/gsoc/pok/trunk/kernel/include/core/debug.h File Reference

Macros

- #define POK_DEBUG_PRINT_CURRENT_STATE
- #define POK_FATAL(arg)

4.86.1 Macro Definition Documentation

4.86.1.1 #define POK_DEBUG_PRINT_CURRENT_STATE

Definition at line 37 of file debug.h.

4.86.1.2 #define POK_FATAL(*arg*)

Definition at line 38 of file debug.h.

4.87 /home/phipse/gsoc/pok/trunk/kernel/include/core/error.h File Reference

4.88 /home/phipse/gsoc/pok/trunk/kernel/include/core/instrumentation.h File Reference

4.89 /home/phipse/gsoc/pok/trunk/kernel/include/core/kernel.h File Reference

Functions

- void pok_kernel_restart (void)
- void pok_kernel_stop (void)

4.89.1 Function Documentation

4.89.1.1 void pok_kernel_restart (void)

4.89.1.2 void pok_kernel_stop (void)

4.90 /home/phipse/gsoc/pok/trunk/kernel/include/core/loader.h File Reference

```
#include <types.h>
```

Functions

- void pok_loader_load_partition (const uint8_t part_id, uint32_t offset, uint32_t *entry)

  Load the program of the partition.

4.90.1 Function Documentation

4.90.1.1 void pok_loader_load_partition (const uint8_t *part_id*, uint32_t *offset*, uint32_t * *entry*)

Load the program of the partition.

It loads the program of the partition *part_id*. In fact, It will load the ELF file that corresponds to this partition.

4.91 /home/phipse/gsoc/pok/trunk/kernel/include/core/lockobj.h File Reference

```
#include <types.h>
#include <arch.h>
```

Data Structures

- struct pok_lockobj_attr_t
- struct pok_lockobj_t
- struct pok_lockobj_lockattr_t

Macros

- #define POK_CONFIG_NB_LOCKOBJECTS 0

Enumerations

- enum pok_lockobj_kind_t { POK_LOCKOBJ_KIND_MUTEX = 1, POK_LOCKOBJ_KIND_SEMAPHORE = 2, POK_LOCKOBJ_KIND_EVENT = 3 }
- enum pok_locking_policy_t { POK_LOCKOBJ_POLICY_STANDARD = 0, POK_LOCKOBJ_POLICY_PIP = 1, POK_LOCKOBJ_POLICY_PCP = 2 }
- enum pok_mutex_state_t { LOCKOBJ_STATE_LOCK = 0, LOCKOBJ_STATE_UNLOCK = 1, LOCKOBJ_STATE_WAITEVENT = 2 }
- enum pok_lockobj_lock_kind_t { LOCKOBK_LOCK_REGULAR = 1, LOCKOBJ_LOCK_TIMED = 2 }
- enum pok_lockobj_operation_t { LOCKOBJ_OPERATION_LOCK = 1, LOCKOBJ_OPERATION_UNLOCK = 2, LOCKOBJ_OPERATION_WAIT = 3, LOCKOBJ_OPERATION_SIGNAL = 4, LOCKOBJ_OPERATION_BROADCAST = 5 }

Functions

- pok_ret_t pok_lockobj_create (pok_lockobj_t *obj, const pok_lockobj_attr_t *attr)
- pok_ret_t pok_lockobj_init ()
- pok_ret_t pok_lockobj_partition_create (pok_lockobj_id_t *id, const pok_lockobj_attr_t *attr)
- pok_ret_t pok_lockobj_lock (pok_lockobj_t *obj, const pok_lockobj_lockattr_t *attr)
- pok_ret_t pok_lockobj_unlock (pok_lockobj_t *obj, const pok_lockobj_lockattr_t *attr)
- pok_ret_t pok_lockobj_eventwait (pok_lockobj_t *obj, const uint64_t timeout)
- pok_ret_t pok_lockobj_eventsignal (pok_lockobj_t *obj)
- pok_ret_t pok_lockobj_eventbroadcast (pok_lockobj_t *obj)
- pok_ret_t pok_lockobj_partition_wrapper (const pok_lockobj_id_t id, const pok_lockobj_lockattr_t *attr)

4.91.1 Macro Definition Documentation

4.91.1.1 #define POK_CONFIG_NB_LOCKOBJECTS 0

Definition at line 25 of file lockobj.h.

4.91.2 Enumeration Type Documentation

4.91.2.1 enum pok_locking_policy_t

Enumerator

POK_LOCKOBJ_POLICY_STANDARD

POK_LOCKOBJ_POLICY_PIP

POK_LOCKOBJ_POLICY_PCP

Definition at line 47 of file lockobj.h.

```
48 {
49    POK_LOCKOBJ_POLICY_STANDARD = 0,
50    POK_LOCKOBJ_POLICY_PIP      = 1,
51    POK_LOCKOBJ_POLICY_PCP      = 2
52 }pok_locking_policy_t;
```

4.91.2.2 enum pok_lockobj_kind_t

Enumerator

POK_LOCKOBJ_KIND_MUTEX

POK_LOCKOBJ_KIND_SEMAPHORE

POK_LOCKOBJ_KIND_EVENT

Definition at line 38 of file lockobj.h.

```
39 {
40    POK_LOCKOBJ_KIND_MUTEX = 1,
41    POK_LOCKOBJ_KIND_SEMAPHORE = 2,
42    POK_LOCKOBJ_KIND_EVENT = 3
43 }pok_lockobj_kind_t;
```

4.91.2.3 enum pok_lockobj_lock_kind_t

Enumerator

LOCKOBK_LOCK_REGULAR

LOCKOBJ_LOCK_TIMED

Definition at line 100 of file lockobj.h.

```
101 {
102    LOCKOBK_LOCK_REGULAR = 1,
103    LOCKOBJ_LOCK_TIMED   = 2
104 }pok_lockobj_lock_kind_t;
```

4.91.2.4 enum pok_lockobj_operation_t

Enumerator

LOCKOBJ_OPERATION_LOCK

LOCKOBJ_OPERATION_UNLOCK

LOCKOBJ_OPERATION_WAIT

LOCKOBJ_OPERATION_SIGNAL

LOCKOBJ_OPERATION_BROADCAST

Definition at line 106 of file lockobj.h.

```
107 {
108    LOCKOBJ_OPERATION_LOCK = 1,
109    LOCKOBJ_OPERATION_UNLOCK = 2,
110    LOCKOBJ_OPERATION_WAIT = 3,
111    LOCKOBJ_OPERATION_SIGNAL = 4,
112    LOCKOBJ_OPERATION_BROADCAST = 5
113 }pok_lockobj_operation_t;
```

4.91.2.5 enum pok_mutex_state_t

Enumerator

LOCKOBJ_STATE_LOCK

LOCKOBJ_STATE_UNLOCK

LOCKOBJ_STATE_WAITEVENT

Definition at line 55 of file lockobj.h.

```
56 {
57    LOCKOBJ_STATE_LOCK = 0,
58    LOCKOBJ_STATE_UNLOCK = 1,
59    LOCKOBJ_STATE_WAITEVENT = 2
60 }pok_mutex_state_t;
```

4.91.3 Function Documentation

4.91.3.1 pok_ret_t pok_lockobj_create (pok_lockobj_t * *obj*, const pok_lockobj_attr_t * *attr*)

4.91.3.2 pok_ret_t pok_lockobj_eventbroadcast (pok_lockobj_t * *obj*)

4.91.3.3 pok_ret_t pok_lockobj_eventsignal (pok_lockobj_t * *obj*)

4.91.3.4 pok_ret_t pok_lockobj_eventwait (pok_lockobj_t * *obj*, const uint64_t *timeout*)

4.91.3.5 pok_ret_t pok_lockobj_init ( )

4.91.3.6 pok_ret_t pok_lockobj_lock (pok_lockobj_t * *obj*, const pok_lockobj_lockattr_t * *attr*)

4.91.3.7 pok_ret_t pok_lockobj_partition_create (pok_lockobj_id_t * *id*, const pok_lockobj_attr_t * *attr*)

4.91.3.8 pok_ret_t pok_lockobj_partition_wrapper (const pok_lockobj_id_t *id*, const pok_lockobj_lockattr_t * *attr*)

4.91.3.9 pok_ret_t pok_lockobj_unlock (pok_lockobj_t * *obj*, const pok_lockobj_lockattr_t * *attr*)

4.92 /home/phipse/gsoc/pok/trunk/kernel/include/core/partition.h File Reference

Definition of structure for partitioning services.

4.92.1 Detailed Description

Definition of structure for partitioning services.

Author

Julien Delange

Definition in file partition.h.

4.93 /home/phipse/gsoc/pok/trunk/kernel/include/core/sched.h File Reference

4.94 /home/phipse/gsoc/pok/trunk/kernel/include/core/schedvalues.h File Reference

Enumerations

- enum pok_sched_t {
  POK_SCHED_FIFO = 0, POK_SCHED_RR = 1, POK_SCHED_GLOBAL_TIMESLICE = 2, POK_SCHED_-
  RMS = 3,
  POK_SCHED_EDF = 4, POK_SCHED_LLF = 5, POK_SCHED_STATIC = 6 }

4.94.1 Enumeration Type Documentation

4.94.1.1 enum pok_sched_t

Enumerator

POK_SCHED_FIFO

POK_SCHED_RR

POK_SCHED_GLOBAL_TIMESLICE

POK_SCHED_RMS

POK_SCHED_EDF

POK_SCHED_LLF

POK_SCHED_STATIC

Definition at line 21 of file schedvalues.h.

```
22 {
23    POK_SCHED_FIFO             = 0,
24    POK_SCHED_RR               = 1,
25    POK_SCHED_GLOBAL_TIMESLICE = 2,
26    POK_SCHED_RMS              = 3,
27    POK_SCHED_EDF              = 4,
28    POK_SCHED_LLF              = 5,
29    POK_SCHED_STATIC           = 6
30 } pok_sched_t;
```

4.95 /home/phipse/gsoc/pok/trunk/kernel/include/core/syscall.h File Reference

```
#include <types.h>
#include <errno.h>
```

Data Structures

- struct pok_syscall_args_t
- struct pok_syscall_info_t

Macros

- #define POK_CHECK_PTR_OR_RETURN(pid, ptr)

Enumerations

- enum pok_syscall_id_t {
  POK_SYSCALL_CONSWRITE = 10, POK_SYSCALL_GETTICK = 20, POK_SYSCALL_INT_NUMBER = 42,
  POK_SYSCALL_THREAD_CREATE = 50,
  POK_SYSCALL_THREAD_SLEEP_UNTIL = 51, POK_SYSCALL_THREAD_SLEEP = 52, POK_SYSCALL-
  _THREAD_SUSPEND = 53, POK_SYSCALL_THREAD_RESTART = 54,
  POK_SYSCALL_THREAD_STOP = 55, POK_SYSCALL_THREAD_PERIOD = 56, POK_SYSCALL_THRE-
  AD_STOPSELF = 57, POK_SYSCALL_THREAD_ID = 58,
  POK_SYSCALL_THREAD_STATUS = 59, POK_SYSCALL_THREAD_SET_PRIORITY = 60, POK_SYSCA-
  LL_THREAD_RESUME = 61, POK_SYSCALL_THREAD_SUSPEND_TARGET = 62,
  POK_SYSCALL_THREAD_DELAYED_START = 65 }

Functions

- pok_ret_t pok_core_syscall (const pok_syscall_id_t syscall_id, const pok_syscall_args_t *args, const pok_-
  syscall_info_t *infos)
- pok_ret_t pok_syscall_init ()

4.95.1 Macro Definition Documentation

4.95.1.1 #define POK_CHECK_PTR_OR_RETURN(*pid, ptr*)

Value:

```
if (!POK_CHECK_PTR_IN_PARTITION(pid,ptr))        \
{                                                \
   return POK_ERRNO_EINVAL;                                \
}
```

Definition at line 143 of file syscall.h.

4.95.2 Enumeration Type Documentation

4.95.2.1 enum pok_syscall_id_t

Enumerator

POK_SYSCALL_CONSWRITE

POK_SYSCALL_GETTICK

POK_SYSCALL_INT_NUMBER

POK_SYSCALL_THREAD_CREATE

POK_SYSCALL_THREAD_SLEEP_UNTIL

POK_SYSCALL_THREAD_SLEEP

POK_SYSCALL_THREAD_SUSPEND

POK_SYSCALL_THREAD_RESTART

POK_SYSCALL_THREAD_STOP

POK_SYSCALL_THREAD_PERIOD

POK_SYSCALL_THREAD_STOPSELF

POK_SYSCALL_THREAD_ID

POK_SYSCALL_THREAD_STATUS

POK_SYSCALL_THREAD_SET_PRIORITY

POK_SYSCALL_THREAD_RESUME

POK_SYSCALL_THREAD_SUSPEND_TARGET

POK_SYSCALL_THREAD_DELAYED_START

Definition at line 23 of file syscall.h.

```
24 {
25    POK_SYSCALL_CONSWRITE                        =  10,
26    POK_SYSCALL_GETTICK                          =  20,
27    POK_SYSCALL_INT_NUMBER                       =  42,
28    POK_SYSCALL_THREAD_CREATE                    =  50,
29    POK_SYSCALL_THREAD_SLEEP_UNTIL               =  51,
30    POK_SYSCALL_THREAD_SLEEP                     =  52,
31    POK_SYSCALL_THREAD_SUSPEND                   =  53,
32    POK_SYSCALL_THREAD_RESTART                   =  54,
33    POK_SYSCALL_THREAD_STOP                      =  55,
34    POK_SYSCALL_THREAD_PERIOD                    =  56,
35    POK_SYSCALL_THREAD_STOPSELF                  =  57,
36    POK_SYSCALL_THREAD_ID                        =  58,
37    POK_SYSCALL_THREAD_STATUS                    =  59,
38    POK_SYSCALL_THREAD_SET_PRIORITY
       =  60,
39    POK_SYSCALL_THREAD_RESUME                    =  61,
40    POK_SYSCALL_THREAD_SUSPEND_TARGET            =  62,
41    //POK_SYSCALL_THREAD_DEADLINE                  =  63,
42    //POK_SYSCALL_THREAD_STATE                     =  64,
43    POK_SYSCALL_THREAD_DELAYED_START             =  65,
44 #ifdef POK_NEEDS_PORTS_SAMPLING
45    POK_SYSCALL_MIDDLEWARE_SAMPLING_ID           = 101,
46    POK_SYSCALL_MIDDLEWARE_SAMPLING_READ         = 102,
47    POK_SYSCALL_MIDDLEWARE_SAMPLING_STATUS       = 103,
48    POK_SYSCALL_MIDDLEWARE_SAMPLING_WRITE        = 104,
49    POK_SYSCALL_MIDDLEWARE_SAMPLING_CREATE       = 105,
50 #endif
51 #ifdef POK_NEEDS_PORTS_QUEUEING
52    POK_SYSCALL_MIDDLEWARE_QUEUEING_CREATE       = 110,
53    POK_SYSCALL_MIDDLEWARE_QUEUEING_SEND         = 111,
54    POK_SYSCALL_MIDDLEWARE_QUEUEING_RECEIVE      = 112,
55    POK_SYSCALL_MIDDLEWARE_QUEUEING_ID           = 113,
56    POK_SYSCALL_MIDDLEWARE_QUEUEING_STATUS       = 114,
57 #endif
58 #ifdef POK_NEEDS_PORTS_VIRTUAL
59    POK_SYSCALL_MIDDLEWARE_VIRTUAL_CREATE        = 150,
60    POK_SYSCALL_MIDDLEWARE_VIRTUAL_NB_DESTINATIONS  = 151,
61    POK_SYSCALL_MIDDLEWARE_VIRTUAL_DESTINATION   = 152,
62    POK_SYSCALL_MIDDLEWARE_VIRTUAL_GET_GLOBAL    = 153,
63 #endif
```

```
64 #if defined (POK_NEEDS_LOCKOBJECTS) || defined (POK_NEEDS_MUTEXES) || defined (POK_NEEDS_SEMAPHORES) ||
       defined (POK_NEEDS_EVENTS) || defined (POK_NEEDS_BUFFERS) || defined (POK_NEEDS_BLACKBOARDS)
65    POK_SYSCALL_LOCKOBJ_CREATE                    = 201,
66    POK_SYSCALL_LOCKOBJ_OPERATION                 = 202,
67 #endif
68 #ifdef POK_NEEDS_ERROR_HANDLING
69    POK_SYSCALL_ERROR_HANDLER_CREATE              = 301,
70    POK_SYSCALL_ERROR_HANDLER_SET_READY           = 302,
71    POK_SYSCALL_ERROR_RAISE_APPLICATION_ERROR     = 303,
72    POK_SYSCALL_ERROR_GET                         = 304,
73 #endif
74 #ifdef POK_NEEDS_PARTITIONS
75    POK_SYSCALL_PARTITION_SET_MODE                = 404,
76    POK_SYSCALL_PARTITION_GET_ID                  = 405,
77    POK_SYSCALL_PARTITION_GET_PERIOD              = 406,
78    POK_SYSCALL_PARTITION_GET_DURATION            = 407,
79    POK_SYSCALL_PARTITION_GET_LOCK_LEVEL          = 408,
80    POK_SYSCALL_PARTITION_GET_OPERATING_MODE      = 409,
81    POK_SYSCALL_PARTITION_GET_START_CONDITION     = 410,
82 #endif
83 #ifdef POK_NEEDS_IO
84    POK_SYSCALL_INB                               = 501,
85    POK_SYSCALL_OUTB                              = 502,
86 #endif
87 #ifdef POK_NEEDS_PCI
88    POK_SYSCALL_PCI_REGISTER                      = 601,
89 #endif
90 } pok_syscall_id_t;
```

4.95.3 Function Documentation

4.95.3.1 pok_ret_t pok_core_syscall (const pok_syscall_id_t *syscall_id*, const pok_syscall_args_t * *args*, const pok_syscall_info_t * *infos*)

Function that performs the syscall. It is called by the architecture interruption handler.

Parameters

syscall_id	This param correspond to the syscal which should be performed. The list of available syscalls is available in the definition of the pok_syscall_id_t type
args	Arguments of the syscall. It corresponds to data useful to perform the syscall.
infos	Informations about the syscall: which partition/thread initiates the syscall, etc ...

Returns

Returns an error code, which is defined in include/errno.h

Here is the default syscall handler. In this case, the syscall ID was not properly identified and thus, we should return an error. If error management is activated, we raise an error in kernel of partitions, calling the error handler.

Definition at line 40 of file syscall.c.

```
43 {
44    switch (syscall_id)
45    {
46 #if defined (POK_NEEDS_CONSOLE) || defined (POK_NEEDS_DEBUG)
47       case POK_SYSCALL_CONSWRITE:
48          POK_CHECK_PTR_OR_RETURN(infos->partition, args->
      arg1 + infos->base_addr)
49          if (pok_cons_write ((const char*)args->arg1 + infos->base_addr, args->arg2))
50          {
51             return POK_ERRNO_OK;
52          }
53          else
54          {
55             return POK_ERRNO_EINVAL;
56          }
57          break;
58 #endif
59
60 #ifdef POK_NEEDS_PORTS_VIRTUAL
61       case POK_SYSCALL_MIDDLEWARE_VIRTUAL_CREATE:
62          POK_CHECK_PTR_OR_RETURN(infos->partition, args->
      arg1 + infos->base_addr)
```

```
63          POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
64          return pok_port_virtual_id ( (char*) (args->arg1 + infos->base_addr), (
    pok_port_id_t*) (args->arg2 + infos->base_addr));
65          break;
66
67       case POK_SYSCALL_MIDDLEWARE_VIRTUAL_NB_DESTINATIONS:
68          POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
69          return pok_port_virtual_nb_destinations ( (pok_port_id_t) (args->arg1), (
    uint32_t*) (args->arg2 + infos->base_addr));
70          break;
71
72       case POK_SYSCALL_MIDDLEWARE_VIRTUAL_DESTINATION:
73          POK_CHECK_PTR_OR_RETURN(infos->partition, ((void*) args->arg3)+infos->
    base_addr)
74          return pok_port_virtual_destination ( (pok_port_id_t) (args->arg1), (
    uint32_t) (args->arg2), (uint32_t*) (args->arg3 + infos->base_addr));
75          break;
76
77       case POK_SYSCALL_MIDDLEWARE_VIRTUAL_GET_GLOBAL:
78          POK_CHECK_PTR_OR_RETURN(infos->partition, (void*) (args->arg2 + infos->
    base_addr))
79          return pok_port_virtual_get_global ((pok_port_id_t) (args->arg1), (
    pok_port_id_t*) (args->arg2 + infos->base_addr));
80          break;
81
82 #endif
83
84 #if defined POK_NEEDS_GETTICK
85       case POK_SYSCALL_GETTICK:
86          POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg1 + infos->base_addr)
87          return pok_gettick_by_pointer ((uint64_t*) (args->arg1 + infos->base_addr));
88          break;
89 #endif
90
91       case POK_SYSCALL_THREAD_CREATE:
92          return pok_partition_thread_create  ((uint32_t*)         (args->
    arg1 + infos->base_addr),
93                                              (pok_thread_attr_t*) (args->arg2 + infos->
    base_addr),
94                                              (uint8_t)            infos->
    partition);
95          break;
96
97 #ifdef POK_NEEDS_THREAD_SLEEP
98       case POK_SYSCALL_THREAD_SLEEP:
99          return pok_thread_sleep (args->arg1);
100          break;
101 #endif
102
103 #ifdef POK_NEEDS_THREAD_SLEEP_UNTIL
104       case POK_SYSCALL_THREAD_SLEEP_UNTIL:
105          return pok_thread_sleep_until (args->arg1);
106          break;
107 #endif
108
109       case POK_SYSCALL_THREAD_PERIOD:
110          return pok_sched_end_period ();
111          break;
112
113 #if defined (POK_NEEDS_THREAD_SUSPEND) || defined (POK_NEEDS_ERROR_HANDLING)
114       case POK_SYSCALL_THREAD_SUSPEND:
115          return pok_thread_suspend ();
116          break;
117 #endif
118
119 #ifdef POK_NEEDS_THREAD_ID
120       case POK_SYSCALL_THREAD_ID:
121          return pok_sched_get_current ((uint32_t*) (args->arg1 + infos->
    base_addr));
122          break;
123 #endif
124    case POK_SYSCALL_THREAD_STATUS:
125             return pok_thread_get_status (args->arg1, (pok_thread_attr_t*) (args->
    arg2 + infos->base_addr));
126          break;
127
128    case POK_SYSCALL_THREAD_DELAYED_START:
129      return pok_thread_delayed_start (args->arg1, args->arg2);
130      break;
131    case POK_SYSCALL_THREAD_SET_PRIORITY:
132           return pok_thread_set_priority (args->arg1, args->arg2);
133           break;
134
135    case POK_SYSCALL_THREAD_RESUME:
136           return pok_thread_resume (args->arg1);
137           break;
```

```
138    case POK_SYSCALL_THREAD_SUSPEND_TARGET:
139            return pok_thread_suspend_target (args->arg1);
140            break;
141
142 #ifdef POK_NEEDS_ERROR_HANDLING
143
148       case POK_SYSCALL_THREAD_RESTART:
149          return pok_partition_restart_thread (args->arg1);
150          break;
151
152       case POK_SYSCALL_THREAD_STOP:
153          return pok_partition_stop_thread (args->arg1);
154          break;
155
159       case POK_SYSCALL_THREAD_STOPSELF:
160          pok_sched_stop_self ();
161          return POK_ERRNO_OK;
162          break;
163
164 #endif
165
166 #ifdef POK_NEEDS_PARTITIONS
167       case POK_SYSCALL_PARTITION_SET_MODE:
168          return pok_partition_set_mode_current ((pok_partition_mode_t)args->arg1);
169          break;
170       case POK_SYSCALL_PARTITION_GET_ID:
171         return pok_current_partition_get_id ((uint8_t*)(args->arg1 + infos->
    base_addr));
172          break;
173       case POK_SYSCALL_PARTITION_GET_PERIOD:
174         return pok_current_partition_get_period ((uint64_t*)(args->arg1 + infos->
    base_addr));
175          break;
176       case POK_SYSCALL_PARTITION_GET_DURATION:
177         return pok_current_partition_get_duration ((uint64_t*)(args->
    arg1 + infos->base_addr));
178          break;
179       case POK_SYSCALL_PARTITION_GET_LOCK_LEVEL:
180         return pok_current_partition_get_lock_level ((uint32_t*)(args->
    arg1 + infos->base_addr));
181          break;
182       case POK_SYSCALL_PARTITION_GET_OPERATING_MODE:
183         return pok_current_partition_get_operating_mode ((pok_partition_mode_t*)(args->
    arg1 + infos->base_addr));
184          break;
185       case POK_SYSCALL_PARTITION_GET_START_CONDITION:
186         return pok_current_partition_get_start_condition ((pok_start_condition_t*)(args->
    arg1 + infos->base_addr));
187          break;
188 #endif
189
190 #ifdef POK_NEEDS_ERROR_HANDLING
191       case POK_SYSCALL_ERROR_HANDLER_CREATE:
192          return pok_error_thread_create (args->arg1 , (void*) (args->arg2));
193          break;
194
195       case POK_SYSCALL_ERROR_RAISE_APPLICATION_ERROR:
196          POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg1 + infos->base_addr)
197          pok_error_raise_application_error ((char*) (args->arg1 + infos->base_addr), args->arg2);
198          return POK_ERRNO_OK;
199          break;
200
201       case POK_SYSCALL_ERROR_GET:
202          return pok_error_get ((pok_error_status_t*) (args->arg1 + infos->base_addr));
203          break;
204 #endif
205
206          /* Middleware syscalls */
207 #ifdef POK_NEEDS_PORTS_SAMPLING
208       case POK_SYSCALL_MIDDLEWARE_SAMPLING_CREATE:
209          POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg5 + infos->base_addr)
210          POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg1 + infos->base_addr)
211          return pok_port_sampling_create  ((char*)(args->arg1 + infos->base_addr),
212                                           (pok_port_size_t) args->arg2,
213                                           (pok_port_direction_t) args->arg3,
214                                           (uint64_t) args->arg4,
215                                           (pok_port_id_t*) (args->arg5 + infos->base_addr));
216          break;
217
218       case POK_SYSCALL_MIDDLEWARE_SAMPLING_WRITE:
219          POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
220
221          return pok_port_sampling_write   ((const pok_port_id_t)args->arg1,
222                                           (const void*) ((void*)args->arg2 + infos->base_addr),
223                                           (const uint8_t) args->arg3);
```

```
224         break;
225
226      case POK_SYSCALL_MIDDLEWARE_SAMPLING_READ:
227         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
228         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg4 + infos->base_addr)
229         return pok_port_sampling_read ((const pok_port_id_t)args->arg1,
230                                        (void*) args->arg2 + infos->base_addr,
231                                        (pok_port_size_t*) (args->arg3 + infos->base_addr),
232                                        (bool_t*) (args->arg4 + infos->base_addr));
233         break;
234
235      case POK_SYSCALL_MIDDLEWARE_SAMPLING_ID:
236         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg1 + infos->base_addr)
237         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
238         return pok_port_sampling_id   ((char*)(args->arg1 + infos->base_addr),
239                                        (pok_port_id_t*)(args->arg2 + infos->base_addr));
240         break;
241
242 #ifndef POK_GENERATED_CODE
243      case POK_SYSCALL_MIDDLEWARE_SAMPLING_STATUS:
244         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg2+infos->base_addr)
245         return pok_port_sampling_status ((const pok_port_id_t)args->arg1,
246                                          (pok_port_sampling_status_t*) (args->arg2 + infos->base_addr));
247         break;
248 #endif /* POK_GENERATED_CODE */
249 #endif /* POK_NEEDS_PORTS_SAMPLING */
250
251
252 #ifdef POK_NEEDS_PORTS_QUEUEING
253      case POK_SYSCALL_MIDDLEWARE_QUEUEING_CREATE:
254         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg1 + infos->base_addr)
255         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg5 + infos->base_addr)
256         return pok_port_queueing_create ((char*)                          (args->arg1 + infos->base_addr),

257                                          (pok_port_size_t)                args->arg2,
258                                          (pok_port_direction_t)           args->arg3,
259                                          (pok_port_queueing_discipline_t)
    args->arg4,
260                                          (pok_port_id_t*)                 (args->arg5 + infos
    ->base_addr));
261         break;
262
263      case POK_SYSCALL_MIDDLEWARE_QUEUEING_SEND:
264         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
265         return pok_port_queueing_send ((const pok_port_id_t)             args->arg1,
266                                        (const void*)                     ((void*)args->arg2 + infos->
    base_addr),
267                                        (const uint8_t)                   (args->arg3),
268                                        (const uint64_t)                  args->arg4);
269         break;
270
271      case POK_SYSCALL_MIDDLEWARE_QUEUEING_RECEIVE:
272         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg4 + infos->base_addr)
273         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg5 + infos->base_addr)
274         return pok_port_queueing_receive ((const pok_port_id_t)   args->arg1,
275                                           (uint64_t)              args->arg2,
276                                           (pok_port_size_t)       args->arg3,
277                                           (void*)                 ((void*)args->arg4 + infos->base_addr),
278                                           (pok_port_size_t*)      (args->arg5 + infos->
    base_addr));
279         break;
280
281      case POK_SYSCALL_MIDDLEWARE_QUEUEING_ID:
282         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg1 + infos->base_addr)
283         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg2 + infos->base_addr)
284         return pok_port_queueing_id ((char*)          (args->arg1 + infos->base_addr),
285                                      (pok_port_id_t*)  (args->arg2 + infos->base_addr));
286         break;
287
288 #ifndef POK_GENERATED_CODE
289      case POK_SYSCALL_MIDDLEWARE_QUEUEING_STATUS:
290         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg2 + infos->base_addr)
291         return pok_port_queueing_status     ((const pok_port_id_t)           args->arg1,
292                                              (pok_port_queueing_status_t*)    (args->arg2 + infos->
    base_addr));
293         break;
294 #endif
295 #endif /* POK_NEEDS_PORTS_QUEUEING */
296
297 #ifdef POK_NEEDS_LOCKOBJECTS
298      case POK_SYSCALL_LOCKOBJ_CREATE:
299         POK_CHECK_PTR_OR_RETURN(infos->partition, args->
    arg2+infos->base_addr)
300         POK_CHECK_PTR_OR_RETURN(infos->partition, args->arg1+infos->base_addr)
```

```
301          return pok_lockobj_partition_create    ((
     pok_lockobj_id_t*)    (args->arg1 + infos->base_addr),
302                                                 (pok_lockobj_attr_t*)   (args->arg2 +
     infos->base_addr));
303          break;
304
305       case POK_SYSCALL_LOCKOBJ_OPERATION:
306          if (args->arg2 == NULL)
307          {
308             return pok_lockobj_partition_wrapper ((const
     uint8_t) args->arg1, NULL);
309          }
310          else
311          {
312             POK_CHECK_PTR_OR_RETURN(infos->partition, args->
     arg2 + infos->base_addr)
313             return pok_lockobj_partition_wrapper   ((const
     uint8_t) args->arg1,
314                                                    (pok_lockobj_lockattr_t*) (args->
     arg2 + infos->base_addr));
315          }
316          break;
317 #endif /* POK_NEEDS_LOCKOBJECTS */
318
319 #ifdef POK_NEEDS_IO
320       case POK_SYSCALL_INB:
321          if ((args->arg1 < pok_partitions[infos->partition].io_min) ||
322              (args->arg1 > pok_partitions[infos->partition].io_max))
323          {
324             return -POK_ERRNO_EPERM;
325          }
326          else
327          {
328             return inb((unsigned short) args->arg1);
329          }
330          break;
331
332       case POK_SYSCALL_OUTB:
333          if ((args->arg1 < pok_partitions[infos->partition].io_min) ||
334              (args->arg1 > pok_partitions[infos->partition].io_max))
335          {
336             return -POK_ERRNO_EPERM;
337          }
338          else
339          {
340             outb((unsigned short) args->arg1, (unsigned char) args->arg2);
341             return POK_ERRNO_OK;
342          }
343        break;
344 #endif /* POK_NEEDS_IO */
345
346 #ifdef POK_NEEDS_PCI
347      case POK_SYSCALL_PCI_REGISTER:
348        POK_CHECK_PTR_OR_RETURN(infos->partition, args->
     arg1 + infos->base_addr)
349        return pci_register((void*)args->arg1 + infos->base_addr, infos->partition);
350        break;
351 #endif /* POK_NEEDS_PCI */
352
353
360       default:
361 #ifdef POK_NEEDS_ERROR_HANDLING
362          pok_error_declare (POK_ERROR_KIND_ILLEGAL_REQUEST);
363          pok_sched_activate_error_thread ();
364 #else
365          #ifdef POK_NEEDS_DEBUG
366             printf ("Tried to use syscall %d\n", syscall_id);
367          #endif
368          POK_FATAL ("Unknown syscall");
369 #endif
370          break;
371    }
372
373    return POK_ERRNO_EINVAL;
374 }
```

4.95.3.2 pok_ret_t pok_syscall_init ( )

Init system calls

Definition at line 83 of file syscalls.c.

```
84 {
```

```
85    pok_idt_set_gate (POK_SYSCALL_INT_NUMBER,
86                      GDT_CORE_CODE_SEGMENT << 3,
87                      (uint32_t) syscall_gate,
88                      IDTE_INTERRUPT,
89                      3);
90
91    return (POK_ERRNO_OK);
92 }
```

4.96 /home/phipse/gsoc/pok/trunk/kernel/include/core/time.h File Reference

4.97 /home/phipse/gsoc/pok/trunk/kernel/include/dependencies.h File Reference

4.98 /home/phipse/gsoc/pok/trunk/kernel/include/elf.h File Reference

Data Structures

- struct Elf32_Ehdr
- struct Elf32_Phdr

Macros

- #define EI_NIDENT (16)

Typedefs

- typedef uint16_t Elf32_Half
- typedef uint32_t Elf32_Word
- typedef uint32_t Elf32_Off
- typedef uint32_t Elf32_Addr

4.98.1 Macro Definition Documentation

4.98.1.1 #define EI_NIDENT (16)

Definition at line 26 of file elf.h.

4.98.2 Typedef Documentation

4.98.2.1 typedef uint32_t Elf32_Addr

Definition at line 24 of file elf.h.

4.98.2.2 typedef uint16_t Elf32_Half

Definition at line 21 of file elf.h.

4.98.2.3 typedef uint32_t Elf32_Off

Definition at line 23 of file elf.h.

4.98.2.4 typedef uint32_t Elf32_Word

Definition at line 22 of file elf.h.

4.99 /home/phipse/gsoc/pok/trunk/kernel/include/errno.h File Reference

Enumerations

- enum pok_ret_t {
  POK_ERRNO_OK = 0, POK_ERRNO_EINVAL = 1, POK_ERRNO_UNAVAILABLE = 2, POK_ERRNO_PARAM = 3,
  POK_ERRNO_TOOMANY = 5, POK_ERRNO_EPERM = 6, POK_ERRNO_EXISTS = 7, POK_ERRNO_ERANGE = 8,
  POK_ERRNO_EDOM = 9, POK_ERRNO_HUGE_VAL = 10, POK_ERRNO_EFAULT = 11, POK_ERRNO_THREAD = 49,
  POK_ERRNO_THREADATTR = 50, POK_ERRNO_TIME = 100, POK_ERRNO_PARTITION_ATTR = 200, POK_ERRNO_PORT = 301,
  POK_ERRNO_NOTFOUND = 302, POK_ERRNO_DIRECTION = 303, POK_ERRNO_SIZE = 304, POK_ERRNO_DISCIPLINE = 305,
  POK_ERRNO_PORTPART = 307, POK_ERRNO_EMPTY = 308, POK_ERRNO_KIND = 309, POK_ERRNO_FULL = 311,
  POK_ERRNO_READY = 310, POK_ERRNO_TIMEOUT = 250, POK_ERRNO_MODE = 251, POK_ERRNO_LOCKOBJ_UNAVAILABLE = 500,
  POK_ERRNO_LOCKOBJ_NOTREADY = 501, POK_ERRNO_LOCKOBJ_KIND = 502, POK_ERRNO_LOCKOBJ_POLICY = 503, POK_ERRNO_PARTITION_MODE = 601,
  POK_ERRNO_PARTITION = 401 }

4.99.1 Enumeration Type Documentation

4.99.1.1 enum pok_ret_t

Enumerator

POK_ERRNO_OK

POK_ERRNO_EINVAL

POK_ERRNO_UNAVAILABLE

POK_ERRNO_PARAM

POK_ERRNO_TOOMANY

POK_ERRNO_EPERM

POK_ERRNO_EXISTS

POK_ERRNO_ERANGE

POK_ERRNO_EDOM

POK_ERRNO_HUGE_VAL

POK_ERRNO_EFAULT

POK_ERRNO_THREAD

POK_ERRNO_THREADATTR

POK_ERRNO_TIME

POK_ERRNO_PARTITION_ATTR

POK_ERRNO_PORT

POK_ERRNO_NOTFOUND

POK_ERRNO_DIRECTION

POK_ERRNO_SIZE
POK_ERRNO_DISCIPLINE
POK_ERRNO_PORTPART
POK_ERRNO_EMPTY
POK_ERRNO_KIND
POK_ERRNO_FULL
POK_ERRNO_READY
POK_ERRNO_TIMEOUT
POK_ERRNO_MODE
POK_ERRNO_LOCKOBJ_UNAVAILABLE
POK_ERRNO_LOCKOBJ_NOTREADY
POK_ERRNO_LOCKOBJ_KIND
POK_ERRNO_LOCKOBJ_POLICY
POK_ERRNO_PARTITION_MODE
POK_ERRNO_PARTITION

Definition at line 21 of file errno.h.

```
22 {
23                 POK_ERRNO_OK                    =   0,
24                 POK_ERRNO_EINVAL                =   1,
25
26                 POK_ERRNO_UNAVAILABLE           =   2,
27                 POK_ERRNO_PARAM
    =   3,
28                 POK_ERRNO_TOOMANY               =   5,
29                 POK_ERRNO_EPERM                 =   6,
30                 POK_ERRNO_EXISTS                =   7,
31
32                 POK_ERRNO_ERANGE                =   8,
33                 POK_ERRNO_EDOM                  =   9,
34                 POK_ERRNO_HUGE_VAL              =  10,
35
36                 POK_ERRNO_EFAULT                =  11,
37
38                 POK_ERRNO_THREAD                =  49,
39                 POK_ERRNO_THREADATTR            =  50,
40
41                 POK_ERRNO_TIME                 =  100,
42
43                 POK_ERRNO_PARTITION_ATTR        = 200,
44
45                 POK_ERRNO_PORT                 =  301,
46                 POK_ERRNO_NOTFOUND             =  302,
47                 POK_ERRNO_DIRECTION            =  303,
48                 POK_ERRNO_SIZE                 =  304,
49                 POK_ERRNO_DISCIPLINE           =  305,
50                 POK_ERRNO_PORTPART             =  307,
51                 POK_ERRNO_EMPTY                =  308,
52                 POK_ERRNO_KIND                 =  309,
53                 POK_ERRNO_FULL                 =  311,
54                 POK_ERRNO_READY                =  310,
55                 POK_ERRNO_TIMEOUT              =  250,
56                 POK_ERRNO_MODE                 =  251,
57
58                 POK_ERRNO_LOCKOBJ_UNAVAILABLE  =  500,
59                 POK_ERRNO_LOCKOBJ_NOTREADY     =  501,
60                 POK_ERRNO_LOCKOBJ_KIND         =  502,
61                 POK_ERRNO_LOCKOBJ_POLICY       =  503,
62
63                 POK_ERRNO_PARTITION_MODE       =  601,
64
65                 POK_ERRNO_PARTITION            =  401
66 } pok_ret_t;
```

4.100 /home/phipse/gsoc/pok/trunk/kernel/include/libc.h File Reference

```
#include <types.h>
```

Functions

- void * memcpy (void *to, const void *from, size_t n)
- void * memset (void *dest, unsigned char val, size_t count)
- int strlen (const char *str)
- int strcmp (const char *s1, const char *s2)
- int strncmp (const char *s1, const char *s2, size_t size)

4.100.1 Function Documentation

4.100.1.1 void* memcpy (void * *to*, const void * *from*, size_t *n*)

Definition at line 20 of file memcpy.c.

```
23 {
24 #ifdef __i386__
25   int d0;
26   int d1;
27   int d2;
28
29   __asm__ __volatile__(
30                        "rep ; movsl\n\t"
31                        "testb $2,%b4\n\t"
32                        "je 1f\n\t"
33                        "movsw\n"
34                        "1:\ttestb $1,%b4\n\t"
35                        "je 2f\n\t"
36                        "movsb\n"
37                        "2:"
38                        : "=&c" (d0), "=&D" (d1), "=&S" (d2)
39                        :"0" (n/4), "q" (n),"1" ((long) to),"2" ((long) from)
40                        : "memory");
41 #else
42   char *cto = (char *)to;
43   const char *cfrom = (const char *)from;
44
45   for (; n > 0; n--)
46   {
47         *cto++ = *cfrom++;
48   }
49 #endif
50   return (to);
51 }
```

4.100.1.2 void* memset (void * *dest*, unsigned char *val*, size_t *count*)

4.100.1.3 int strcmp (const char * *s1*, const char * *s2*)

4.100.1.4 int strlen (const char * *str*)

4.100.1.5 int strncmp (const char * *s1*, const char * *s2*, size_t *size*)

4.101 /home/phipse/gsoc/pok/trunk/kernel/include/middleware/port.h File Reference

Describe queueing and sampling ports structures.

```
#include <types.h>
#include <errno.h>
#include <core/lockobj.h>
```

Data Structures

- struct pok_port_t

Macros

- #define POK_PORT_MAX_SIZE 512

Typedefs

- typedef pok_queueing_discipline_t pok_port_queueing_discipline_t

Enumerations

- enum pok_port_queueing_disciplines_t { POK_PORT_QUEUEING_DISCIPLINE_FIFO = 1, POK_PORT_QUEUEING_DISCIPLINE_PRIORITY = 2 }
- enum pok_port_directions_t { POK_PORT_DIRECTION_IN = 1, POK_PORT_DIRECTION_OUT = 2 }
- enum pok_port_kinds_t { POK_PORT_KIND_QUEUEING = 1, POK_PORT_KIND_SAMPLING = 2, POK_PORT_KIND_INVALID = 10 }

4.101.1 Detailed Description

Describe queueing and sampling ports structures.

Date

2008-2009

Author

Julien Delange

Definition in file port.h.

4.101.2 Macro Definition Documentation

4.101.2.1 #define POK_PORT_MAX_SIZE 512

Definition at line 31 of file port.h.

4.101.3 Typedef Documentation

4.101.3.1 typedef pok_queueing_discipline_t pok_port_queueing_discipline_t

Definition at line 45 of file port.h.

4.101.4 Enumeration Type Documentation

4.101.4.1 enum pok_port_directions_t

Enumerator

POK_PORT_DIRECTION_IN

POK_PORT_DIRECTION_OUT

Definition at line 39 of file port.h.

```
40 {
41          POK_PORT_DIRECTION_IN    = 1,
42          POK_PORT_DIRECTION_OUT   = 2
43 } pok_port_directions_t;
```

4.101.4.2 enum pok_port_kinds_t

Enumerator

POK_PORT_KIND_QUEUEING

POK_PORT_KIND_SAMPLING

POK_PORT_KIND_INVALID

Definition at line 47 of file port.h.

```
48 {
49          POK_PORT_KIND_QUEUEING  = 1,
50          POK_PORT_KIND_SAMPLING  = 2,
51 #ifdef POK_NEEDS_PORTS_VIRTUAL
52          POK_PORT_KIND_VIRTUAL   = 2,
53 #endif
54          POK_PORT_KIND_INVALID   = 10
55 } pok_port_kinds_t;
```

4.101.4.3 enum pok_port_queueing_disciplines_t

Enumerator

POK_PORT_QUEUEING_DISCIPLINE_FIFO

POK_PORT_QUEUEING_DISCIPLINE_PRIORITY

Definition at line 33 of file port.h.

```
34 {
35          POK_PORT_QUEUEING_DISCIPLINE_FIFO      = 1,
36          POK_PORT_QUEUEING_DISCIPLINE_PRIORITY  = 2
37 } pok_port_queueing_disciplines_t;
```

4.102 /home/phipse/gsoc/pok/trunk/kernel/include/middleware/queue.h File Reference

4.103 /home/phipse/gsoc/pok/trunk/kernel/libc/__udivdi3.c File Reference

Functions

- unsigned long long __udivdi3 (unsigned long long num, unsigned long long den)

4.103.1 Function Documentation

4.103.1.1 unsigned long long __udivdi3 (unsigned long long *num,* unsigned long long *den*)

Definition at line 19 of file __udivdi3.c.

```
21 {
22 #ifdef POK_NEEDS_DEBUG
23    unsigned long long quot, qbit;
24
25    quot = 0;
26    qbit = 1;
```

```
27
28    if (den == 0)
29    {
30       return 0;
31    }
32
33    while ((long long) den >= 0)
34    {
35       den <<= 1;
36       qbit <<= 1;
37    }
38
39    while (qbit)
40    {
41       if (den <= num)
42       {
43          num -= den;
44          quot += qbit;
45       }
46       den >>= 1;
47       qbit >>= 1;
48    }
49
50    return quot;
51 #else
52    (void) num;
53    (void) den;
54    return 0;
55 #endif
56 }
```

4.104 /home/phipse/gsoc/pok/trunk/kernel/libc/memcpy.c File Reference

```
#include <libc.h>
```

Functions

- void * memcpy (void *to, const void *from, size_t n)

4.104.1 Function Documentation

4.104.1.1 void* memcpy (void * *to*, const void * *from*, size_t *n*)

Definition at line 20 of file memcpy.c.

```
23 {
24 #ifdef __i386__
25   int d0;
26   int d1;
27   int d2;
28
29   __asm__ __volatile__(
30                        "rep ; movsl\n\t"
31                        "testb $2,%b4\n\t"
32                        "je 1f\n\t"
33                        "movsw\n"
34                        "1:\ttestb $1,%b4\n\t"
35                        "je 2f\n\t"
36                        "movsb\n"
37                        "2:"
38                        : "=&c" (d0), "=&D" (d1), "=&S" (d2)
39                        :"0" (n/4), "q" (n),"1" ((long) to),"2" ((long) from)
40                        : "memory");
41 #else
42   char *cto = (char *)to;
43   const char *cfrom = (const char *)from;
44
45   for (; n > 0; n--)
46   {
47        *cto++ = *cfrom++;
48   }
49 #endif
50   return (to);
```

```
51 }
```

4.105 /home/phipse/gsoc/pok/trunk/kernel/libc/memset.c File Reference

```
#include <libc.h>
```

Functions

- __attribute__ ((weak))

4.105.1 Function Documentation

4.105.1.1 __attribute__ ((weak))

Definition at line 20 of file memset.c.

```
22 {
23   unsigned char *d = (unsigned char *) dest;
24
25   while (count--)
26   {
27     *d++ = val;
28   }
29
30   return dest;
31 }
```

4.106 /home/phipse/gsoc/pok/trunk/kernel/libc/printf.c File Reference

4.107 /home/phipse/gsoc/pok/trunk/kernel/libc/strcmp.c File Reference

4.108 /home/phipse/gsoc/pok/trunk/kernel/libc/strlen.c File Reference

4.109 /home/phipse/gsoc/pok/trunk/kernel/middleware/portcreate.c File Reference

4.110 /home/phipse/gsoc/pok/trunk/kernel/middleware/portflushall.c File Reference

Flush the ports and send the data of IN ports to OUT ports.

4.110.1 Detailed Description

Flush the ports and send the data of IN ports to OUT ports.

Date

2008-2009

Author

Julien Delange
Laurent Lec

Definition in file portflushall.c.

4.111 /home/phipse/gsoc/pok/trunk/kernel/middleware/portinit.c File Reference

4.112 /home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingcreate.c File Reference

4.113 /home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingid.c File Reference

4.114 /home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingreceive.c File Reference

4.115 /home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingsend.c File Reference

4.116 /home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingstatus.c File Reference

4.117 /home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingcreate.c File Reference

4.118 /home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingid.c File Reference

4.119 /home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingread.c File Reference

4.120 /home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingstatus.c File Reference

4.121 /home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingwrite.c File Reference

Send data on a sampling port.

4.121.1 Detailed Description

Send data on a sampling port.

Author

Julien Delange

Date

2008-2009

Definition in file portsamplingwrite.c.

4.122 /home/phipse/gsoc/pok/trunk/kernel/middleware/portutils.c File Reference

Various functions for ports management.

4.122.1 Detailed Description

Various functions for ports management.

Date

2008-2009

Author

Julien Delange

Definition in file portutils.c.

4.123 /home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualdestination.c File Reference

4.124 /home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualgetglobal.c File Reference

4.125 /home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualid.c File Reference

4.126 /home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualnbdestinations.c File Reference

4.127 /home/phipse/gsoc/pok/trunk/kernel/middleware/queueinit.c File Reference

4.128 /home/phipse/gsoc/pok/trunk/kernel/middleware/ressources.c File Reference

Index

/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/arch.c, 43
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/msr.h, 49
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/bsp.c, 50
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/cons.c, 53
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/cons.h, 55
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/prep/ioports.h, 59
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/space.c, 61
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/syscalls.c, 72
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.c, 76
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/thread.h, 77
/home/phipse/gsoc/pok/trunk/kernel/arch/ppc/timer.c, 79
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/arch.c, 45
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/context_offset.h, 81
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/bsp.c, 51
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/cons.c, 54
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/cons.h, 55
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/ioports.h, 59
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/irq.h, 84
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/sparc_conf.h, 85
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/timer.c, 80
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/leon3/timer.h, 87
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/psr.h, 89
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.c, 65
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/space.h, 90
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/syscalls.c, 73
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/syscalls.h, 95
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/thread.c, 76
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/thread.h, 77
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/traps.c, 95
/home/phipse/gsoc/pok/trunk/kernel/arch/sparc/traps.h, 97
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/arch.c, 47
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.c, 98
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/event.h, 100
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/exceptions.c, 104
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.c, 104
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/gdt.h, 107
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/interrupt.c, 111
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/pci.c, 111
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.c, 69
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/space.h, 94
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/syscalls.c, 74
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/sysdesc.h, 111
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.c, 76
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/thread.h, 78
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/tss.h, 112
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/types.h, 112
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/bsp.c, 52
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/cons.c, 54
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/cons.h, 58
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/debug.c, 118
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pic.c, 118
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pic.h, 119

/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pit.c, 122
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pit.h, 123
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pm.c, 123
/home/phipse/gsoc/pok/trunk/kernel/arch/x86/x86-qemu/pm.h, 125
/home/phipse/gsoc/pok/trunk/kernel/core/boot.c, 126
/home/phipse/gsoc/pok/trunk/kernel/core/cons.c, 55
/home/phipse/gsoc/pok/trunk/kernel/core/debug.c, 118
/home/phipse/gsoc/pok/trunk/kernel/core/error.c, 128
/home/phipse/gsoc/pok/trunk/kernel/core/instrumentation.-c, 128
/home/phipse/gsoc/pok/trunk/kernel/core/kernel.c, 128
/home/phipse/gsoc/pok/trunk/kernel/core/loader.c, 128
/home/phipse/gsoc/pok/trunk/kernel/core/lockobj.c, 128
/home/phipse/gsoc/pok/trunk/kernel/core/partition.c, 128
/home/phipse/gsoc/pok/trunk/kernel/core/sched.c, 129
/home/phipse/gsoc/pok/trunk/kernel/core/syscall.c, 129
/home/phipse/gsoc/pok/trunk/kernel/core/thread.c, 76
/home/phipse/gsoc/pok/trunk/kernel/core/time.c, 134
/home/phipse/gsoc/pok/trunk/kernel/include/arch.h, 134
/home/phipse/gsoc/pok/trunk/kernel/include/arch/ppc/spinlock.-h, 139
/home/phipse/gsoc/pok/trunk/kernel/include/arch/sparc/spinlock.-h, 140
/home/phipse/gsoc/pok/trunk/kernel/include/arch/sparc/types.-h, 113
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/interrupt.-h, 141
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/ioports.-h, 59
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/multiboot.-h, 143
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/pci.-h, 145
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/spinlock.-h, 140
/home/phipse/gsoc/pok/trunk/kernel/include/arch/x86/types.-h, 114
/home/phipse/gsoc/pok/trunk/kernel/include/bsp.h, 145
/home/phipse/gsoc/pok/trunk/kernel/include/core/boot.-h, 147
/home/phipse/gsoc/pok/trunk/kernel/include/core/cons.-h, 59
/home/phipse/gsoc/pok/trunk/kernel/include/core/cpio.h, 148
/home/phipse/gsoc/pok/trunk/kernel/include/core/debug.-h, 149
/home/phipse/gsoc/pok/trunk/kernel/include/core/error.-h, 149
/home/phipse/gsoc/pok/trunk/kernel/include/core/instrumentation.-h, 149
/home/phipse/gsoc/pok/trunk/kernel/include/core/kernel.-h, 149
/home/phipse/gsoc/pok/trunk/kernel/include/core/loader.-h, 150
/home/phipse/gsoc/pok/trunk/kernel/include/core/lockobj.-h, 150
/home/phipse/gsoc/pok/trunk/kernel/include/core/partition.-h, 153
/home/phipse/gsoc/pok/trunk/kernel/include/core/sched.-h, 153
/home/phipse/gsoc/pok/trunk/kernel/include/core/schedvalues.-h, 153
/home/phipse/gsoc/pok/trunk/kernel/include/core/syscall.-h, 154
/home/phipse/gsoc/pok/trunk/kernel/include/core/thread.-h, 79
/home/phipse/gsoc/pok/trunk/kernel/include/core/time.-h, 161
/home/phipse/gsoc/pok/trunk/kernel/include/dependencies.-h, 161
/home/phipse/gsoc/pok/trunk/kernel/include/elf.h, 161
/home/phipse/gsoc/pok/trunk/kernel/include/errno.h, 162
/home/phipse/gsoc/pok/trunk/kernel/include/libc.h, 163
/home/phipse/gsoc/pok/trunk/kernel/include/middleware/port.-h, 164
/home/phipse/gsoc/pok/trunk/kernel/include/middleware/queue.-h, 166
/home/phipse/gsoc/pok/trunk/kernel/include/types.h, 115
/home/phipse/gsoc/pok/trunk/kernel/libc/__udivdi3.c, 166
/home/phipse/gsoc/pok/trunk/kernel/libc/memcpy.c, 167
/home/phipse/gsoc/pok/trunk/kernel/libc/memset.c, 168
/home/phipse/gsoc/pok/trunk/kernel/libc/printf.c, 168
/home/phipse/gsoc/pok/trunk/kernel/libc/strcmp.c, 168
/home/phipse/gsoc/pok/trunk/kernel/libc/strlen.c, 168
/home/phipse/gsoc/pok/trunk/kernel/middleware/portcreate.-c, 168
/home/phipse/gsoc/pok/trunk/kernel/middleware/portflushall.-c, 168
/home/phipse/gsoc/pok/trunk/kernel/middleware/portinit.-c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingcreate.-c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingid.-c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingreceive.-c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingsend.-c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portqueueingstatus.-c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingcreate.-c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingid.-c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingread.-c, 169

/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingstatus.c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portsamplingwstatus.c, 169
/home/phipse/gsoc/pok/trunk/kernel/middleware/portutils.c, 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualdestination.c, 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualgetglobal.c, 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualid.c, 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/portvirtualnbdestinations.c, 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/queueinit.c, 170
/home/phipse/gsoc/pok/trunk/kernel/middleware/ressources.c, 170
__attribute__, 7
  available, 8
  back_link, 8
  base, 8
  base_high, 8
  base_low, 8
  cr3, 8
  cs, 8
  d, 8
  dpl, 9
  ds, 9
  eax, 9
  ebp, 9
  ebx, 9
  ecx, 9
  edi, 9
  edx, 9
  eflags, 9
  eip, 9
  es, 9
  esi, 9
  esp, 10
  esp0, 10
  esp1, 10
  esp2, 10
  fs, 10
  granularity, 10
  gs, 10
  io_bit_map_offset, 10
  ldt, 10
  limit, 10
  limit_high, 10
  limit_low, 10
  memset.c, 168
  offset_high, 11
  offset_low, 11
  op_size, 11
  padding, 11
  present, 11
  res0, 11
  res1, 11
  s, 11
  segsel, 11
  sparc/space.c, 67
  ss, 11
  ss0, 11
  ss1, 11
  ss2, 12
  trap, 12
  type, 12
__esp
  context_t, 13
  interrupt_frame, 22
__pok_begin
  pm.c, 125
__pok_end
  pm.c, 125
__udivdi3
  __udivdi3.c, 166
__udivdi3.c
  __udivdi3, 166
_end
  ppc/prep/bsp.c, 51
  sparc/leon3/bsp.c, 52

ALIGN_UP
  pm.c, 124
ASI_M_MMUREGS
  sparc/space.h, 91
ASI_MMU_BYPASS
  sparc_conf.h, 86
ack_irq
  irq.h, 85
addr
  pok_aout_symbol_table_t, 24
  pok_elf_section_header_table_t, 25
aout_sym
  pok_multiboot_info_t, 31
arch.h
  pok_arch_event_register, 135
  pok_arch_idle, 135
  pok_arch_init, 135
  pok_arch_preempt_disable, 136
  pok_arch_preempt_enable, 136
  pok_context_create, 136
  pok_context_reset, 136
  pok_context_switch, 136
  pok_create_space, 136
  pok_dispatch_space, 137
  pok_space_base_vaddr, 137
  pok_space_context_create, 137
  pok_space_context_restart, 138
  pok_space_switch, 138
  pok_thread_stack_addr, 138
arch/ppc/prep/cons.c
  pok_cons_init, 53
arch/ppc/prep/cons.h
  pok_cons_init, 55
arch/ppc/prep/ioports.h

inb, 59
outb, 59
POK_PREP_IOBASE, 59
arch/ppc/thread.h
pok_context_create, 77
pok_context_switch, 77
arch/sparc/leon3/cons.c
pok_cons_init, 54
arch/sparc/leon3/cons.h
pok_cons_init, 58
UART1, 56
UART_CTRL_FL, 56
UART_CTRL_LB, 56
UART_CTRL_OFFSET, 56
UART_CTRL_PE, 56
UART_CTRL_PS, 56
UART_CTRL_RE, 56
UART_CTRL_RI, 56
UART_CTRL_TE, 57
UART_CTRL_TI, 57
UART_DATA_OFFSET, 57
UART_SCALER_OFFSET, 57
UART_STAT_OFFSET, 57
UART_STATUS_BR, 57
UART_STATUS_DR, 57
UART_STATUS_ERR, 57
UART_STATUS_FE, 57
UART_STATUS_OE, 58
UART_STATUS_PE, 58
UART_STATUS_THE, 58
UART_STATUS_TSE, 58
arch/sparc/thread.h
pok_arch_sp, 78
pok_context_create, 78
pok_context_switch, 78
arch/x86/thread.h
pok_context_create, 78
pok_context_reset, 78
pok_context_switch, 78
arch/x86/types.h
int16_t, 112
int64_t, 112
int8_t, 112
intptr_t, 112
size_t, 112
uint16_t, 113
uint32_t, 113
uint64_t, 113
uint8_t, 113
arch/x86/x86-qemu/cons.c
pok_cons_init, 54
arch/x86/x86-qemu/cons.h
pok_cons_init, 58
arg1
pok_syscall_args_t, 35
space_context_t, 37
arg2
pok_syscall_args_t, 35
space_context_t, 37
arg3
pok_syscall_args_t, 35
arg4
pok_syscall_args_t, 36
arg5
pok_syscall_args_t, 36
available
__attribute__, 8

BUS_FREQ
ppc/timer.c, 79
back_chain
context_t, 13
volatile_context_t, 40
back_link
__attribute__, 8
base
__attribute__, 8
base_addr
pok_syscall_info_t, 36
base_addr_high
pok_memory_map_t, 28
base_addr_low
pok_memory_map_t, 28
base_high
__attribute__, 8
base_low
__attribute__, 8
bool_t
include/types.h, 116
boot.c
pok_boot, 127
boot.h
pok_boot, 147
boot_device
pok_multiboot_info_t, 31
bsp.h
pok_bsp_init, 145
pok_bsp_irq_acknowledge, 145
pok_bsp_irq_register, 146
pok_bsp_mem_alloc, 146
pok_bsp_time_init, 146
pok_cons_write, 146
bss_end_addr
pok_multiboot_header_t, 30

CPIO_BIN_FMT
cpio.h, 148
CPIO_CRC_FMT
cpio.h, 148
CPIO_HPBIN_FMT
cpio.h, 148
CPIO_HPODC_FMT
cpio.h, 148
CPIO_NEWC_FMT
cpio.h, 148
CPIO_ODC_FMT
cpio.h, 148

CPIO_TAR_FMT
cpio.h, 148
CPIO_USTAR_FMT
cpio.h, 148
c_dev
cpio_bin_header, 17
c_filesize
cpio_bin_header, 17
c_gid
cpio_bin_header, 17
c_ino
cpio_bin_header, 17
c_magic
cpio_bin_header, 17
c_mode
cpio_bin_header, 17
c_mtime
cpio_bin_header, 17
c_namesize
cpio_bin_header, 17
c_nlink
cpio_bin_header, 17
c_rdev
cpio_bin_header, 17
c_uid
cpio_bin_header, 17
checksum
pok_multiboot_header_t, 30
cmdline
pok_multiboot_info_t, 31
context_offset.h
G1_OFFSET, 82
G2_OFFSET, 82
G3_OFFSET, 82
G4_OFFSET, 82
G5_OFFSET, 82
G6_OFFSET, 82
G7_OFFSET, 82
I0_OFFSET, 82
I1_OFFSET, 82
I2_OFFSET, 82
I3_OFFSET, 82
I4_OFFSET, 83
I5_OFFSET, 83
I6_OFFSET, 83
I7_OFFSET, 83
L0_OFFSET, 83
L1_OFFSET, 83
L2_OFFSET, 83
L3_OFFSET, 83
L4_OFFSET, 83
L5_OFFSET, 83
L6_OFFSET, 83
L7_OFFSET, 83
NPC_OFFSET, 84
PC_OFFSET, 84
PSR_OFFSET, 84
WIM_OFFSET, 84
Y_OFFSET, 84
context_t, 12
__esp, 13
back_chain, 13
cr, 13
cs, 13
eax, 13
ebp, 13
ebx, 13
ecx, 13
edi, 14
edx, 14
eflags, 14
eip, 14
esi, 14
lr, 14
pad, 14
r13, 14
r14, 14
r15, 14
r16, 14
r17, 14
r18, 15
r19, 15
r2, 15
r20, 15
r21, 15
r22, 15
r23, 15
r24, 15
r25, 15
r26, 15
r27, 15
r28, 15
r29, 16
r30, 16
r31, 16
sp, 16
unused_lr, 16
cpio.h
CPIO_BIN_FMT, 148
CPIO_CRC_FMT, 148
CPIO_HPBIN_FMT, 148
CPIO_HPODC_FMT, 148
CPIO_NEWC_FMT, 148
CPIO_ODC_FMT, 148
CPIO_TAR_FMT, 148
CPIO_USTAR_FMT, 148
cpio.h
cpio_format, 148
cpio_get_fileaddr, 149
cpio_get_filename, 149
cpio_next_file, 149
cpio_open, 149
cpio_addr
cpio_file, 18
cpio_bin_header, 16
c_dev, 17

c_filesize, 17
c_gid, 17
c_ino, 17
c_magic, 17
c_mode, 17
c_mtime, 17
c_namesize, 17
c_nlink, 17
c_rdev, 17
c_uid, 17
cpio_file, 18
cpio_addr, 18
cpio_fmt, 18
curr_fileaddr, 18
curr_filename, 18
curr_filename_len, 18
curr_filesz, 18
curr_header, 18
next_header, 18
cpio_fmt
cpio_file, 18
cpio_format
cpio.h, 148
cpio_get_fileaddr
cpio.h, 149
cpio_get_filename
cpio.h, 149
cpio_next_file
cpio.h, 149
cpio_open
cpio.h, 149
cr
context_t, 13
volatile_context_t, 40
cr3
__attribute__, 8
cs
__attribute__, 8
context_t, 13
interrupt_frame, 22
ctr
volatile_context_t, 40
ctx
space_context_t, 38
start_context_t, 39
curr_fileaddr
cpio_file, 18
curr_filename
cpio_file, 18
curr_filename_len
cpio_file, 18
curr_filesz
cpio_file, 18
curr_header
cpio_file, 18
current_value
pok_lockobj_t, 27

d
__attribute__, 8
debug.h
POK_FATAL, 149
direction
pok_port_t, 33
discipline
pok_port_t, 33
dpl
__attribute__, 9
ds
__attribute__, 9
interrupt_frame, 22

e_ehsize
Elf32_Ehdr, 19
e_entry
Elf32_Ehdr, 19
e_flags
Elf32_Ehdr, 19
e_gdte_type
gdt.h, 109
e_ident
Elf32_Ehdr, 19
e_idte_type
event.h, 102, 103
e_machine
Elf32_Ehdr, 19
e_phentsize
Elf32_Ehdr, 20
e_phnum
Elf32_Ehdr, 20
e_phoff
Elf32_Ehdr, 20
e_shentsize
Elf32_Ehdr, 20
e_shnum
Elf32_Ehdr, 20
e_shoff
Elf32_Ehdr, 20
e_shstrndx
Elf32_Ehdr, 20
e_type
Elf32_Ehdr, 20
e_version
Elf32_Ehdr, 20
EI_NIDENT
elf.h, 161
EXCEPTION_BOUNDRANGE
event.h, 101
EXCEPTION_BREAKPOINT
event.h, 101
EXCEPTION_DEBUG
event.h, 101
EXCEPTION_DOUBLEFAULT
event.h, 101
EXCEPTION_FPU_FAULT
event.h, 101
EXCEPTION_NMI
event.h, 102

EXCEPTION_OVERFLOW
event.h, 102
EXCEPTION_PAGEFAULT
event.h, 102
EXCEPTION_RESERVED
event.h, 102
EXCEPTION_SIMD_FAULT
event.h, 102
EXT_C
multiboot.h, 144
eax
__attribute__, 9
context_t, 13
interrupt_frame, 22
ebp
__attribute__, 9
context_t, 13
interrupt_frame, 22
ebx
__attribute__, 9
context_t, 13
interrupt_frame, 22
ecx
__attribute__, 9
context_t, 13
interrupt_frame, 23
edi
__attribute__, 9
context_t, 14
interrupt_frame, 23
edx
__attribute__, 9
context_t, 14
interrupt_frame, 23
eflags
__attribute__, 9
context_t, 14
interrupt_frame, 23
eip
__attribute__, 9
context_t, 14
interrupt_frame, 23
elf.h
EI_NIDENT, 161
Elf32_Addr, 161
Elf32_Half, 161
Elf32_Off, 161
Elf32_Word, 161
Elf32_Addr
elf.h, 161
Elf32_Ehdr, 19
e_ehsize, 19
e_entry, 19
e_flags, 19
e_ident, 19
e_machine, 19
e_phentsize, 20
e_phnum, 20
e_phoff, 20
e_shentsize, 20
e_shnum, 20
e_shoff, 20
e_shstrndx, 20
e_type, 20
e_version, 20
Elf32_Half
elf.h, 161
Elf32_Off
elf.h, 161
Elf32_Phdr, 20
p_align, 21
p_filesz, 21
p_flags, 21
p_memsz, 21
p_offset, 21
p_paddr, 21
p_type, 21
p_vaddr, 21
Elf32_Word
elf.h, 161
elf_sec
pok_multiboot_info_t, 32
empty
pok_port_t, 33
entry
start_context_t, 39
entry_addr
pok_multiboot_header_t, 30
errno.h
POK_ERRNO_DIRECTION, 162
POK_ERRNO_DISCIPLINE, 163
POK_ERRNO_EDOM, 162
POK_ERRNO_EFAULT, 162
POK_ERRNO_EINVAL, 162
POK_ERRNO_EMPTY, 163
POK_ERRNO_EPERM, 162
POK_ERRNO_ERANGE, 162
POK_ERRNO_EXISTS, 162
POK_ERRNO_FULL, 163
POK_ERRNO_HUGE_VAL, 162
POK_ERRNO_KIND, 163
POK_ERRNO_LOCKOBJ_KIND, 163
POK_ERRNO_LOCKOBJ_NOTREADY, 163
POK_ERRNO_LOCKOBJ_POLICY, 163
POK_ERRNO_LOCKOBJ_UNAVAILABLE, 163
POK_ERRNO_MODE, 163
POK_ERRNO_NOTFOUND, 162
POK_ERRNO_OK, 162
POK_ERRNO_PARAM, 162
POK_ERRNO_PARTITION, 163
POK_ERRNO_PARTITION_ATTR, 162
POK_ERRNO_PARTITION_MODE, 163
POK_ERRNO_PORT, 162
POK_ERRNO_PORTPART, 163
POK_ERRNO_READY, 163
POK_ERRNO_SIZE, 162

POK_ERRNO_THREAD, 162
POK_ERRNO_THREADATTR, 162
POK_ERRNO_TIME, 162
POK_ERRNO_TIMEOUT, 163
POK_ERRNO_TOOMANY, 162
POK_ERRNO_UNAVAILABLE, 162
errno.h
pok_ret_t, 162
error
interrupt_frame, 23
es
__attribute__, 9
interrupt_frame, 23
esi
__attribute__, 9
context_t, 14
interrupt_frame, 23
esp
__attribute__, 10
interrupt_frame, 23
esp0
__attribute__, 10
esp1
__attribute__, 10
esp2
__attribute__, 10
event.h
IDTE_INTERRUPT, 103
IDTE_TASK, 103
IDTE_TRAP, 103
event.c
IDT_SIZE, 99
pok_event_init, 99
pok_idt, 100
pok_idt_init, 99
pok_idt_set_gate, 99
event.h
e_idte_type, 102, 103
EXCEPTION_BOUNDRANGE, 101
EXCEPTION_BREAKPOINT, 101
EXCEPTION_DEBUG, 101
EXCEPTION_FPU_FAULT, 101
EXCEPTION_NMI, 102
EXCEPTION_OVERFLOW, 102
EXCEPTION_PAGEFAULT, 102
EXCEPTION_RESERVED, 102
pok_event_init, 103
pok_exception_init, 103
pok_idt_init, 103
pok_idt_set_gate, 103
pok_syscall_init, 104
eventspin
pok_lockobj_t, 27

FALSE
include/types.h, 116
FREQ_DIV
ppc/timer.c, 79
fake_ret
space_context_t, 38
start_context_t, 39
flags
pok_multiboot_header_t, 30
pok_multiboot_info_t, 32
fs
__attribute__, 10
full
pok_port_t, 33

G1_OFFSET
context_offset.h, 82
G2_OFFSET
context_offset.h, 82
G3_OFFSET
context_offset.h, 82
G4_OFFSET
context_offset.h, 82
G5_OFFSET
context_offset.h, 82
G6_OFFSET
context_offset.h, 82
G7_OFFSET
context_offset.h, 82
GDTE_CODE
gdt.h, 109
GDTE_DATA
gdt.h, 109
GDTE_TSS
gdt.h, 109
GDT_BUILD_SELECTOR
gdt.h, 108
GDT_SIZE
gdt.c, 105
GDT_TSS_SEGMENT
gdt.h, 108
gdt.h
GDTE_CODE, 109
GDTE_DATA, 109
GDTE_TSS, 109
gdt.c
GDT_SIZE, 105
gdt_disable, 105
gdt_enable, 105
gdt_set_segment, 105
gdt_set_system, 106
pok_gdt, 107
pok_gdt_init, 106
pok_tss, 107
pok_tss_init, 107
tss_set_esp0, 107
gdt.h
e_gdte_type, 109
GDT_BUILD_SELECTOR, 108
GDT_TSS_SEGMENT, 108
gdt_disable, 109
gdt_enable, 109
gdt_set_segment, 109
gdt_set_system, 110

pok_gdt_init, 110
pok_tss_init, 110
tss_set_esp0, 111
gdt_disable
gdt.c, 105
gdt.h, 109
gdt_enable
gdt.c, 105
gdt.h, 109
gdt_set_segment
gdt.c, 105
gdt.h, 109
gdt_set_system
gdt.c, 106
gdt.h, 110
granularity
__attribute__, 10
gs
__attribute__, 10

header_addr
pok_multiboot_header_t, 30

I0_OFFSET
context_offset.h, 82
I1_OFFSET
context_offset.h, 82
I2_OFFSET
context_offset.h, 82
I3_OFFSET
context_offset.h, 82
I4_OFFSET
context_offset.h, 83
I5_OFFSET
context_offset.h, 83
I6_OFFSET
context_offset.h, 83
I7_OFFSET
context_offset.h, 83
IDTE_INTERRUPT
event.h, 103
IDTE_TASK
event.h, 103
IDTE_TRAP
event.h, 103
IDT_SIZE
event.c, 99
INTERRUPT_HANDLER
interrupt.h, 142
pit.c, 123
INTERRUPT_HANDLER_errorcode
interrupt.h, 142
INTERRUPT_HANDLER_syscall
interrupt.h, 142
x86/syscalls.c, 75
IRQMP_BASE
irq.h, 85
IRQMP_CLEAR_OFFSET
irq.h, 85
IRQMP_MASK0_OFFSET
irq.h, 85
id
start_context_t, 39
identifier
pok_port_t, 33
inb
arch/ppc/prep/ioports.h, 59
include/arch/x86/ioports.h, 60
include/arch/sparc/types.h
int16_t, 113
int64_t, 113
int8_t, 113
intptr_t, 113
size_t, 114
uint16_t, 114
uint32_t, 114
uint64_t, 114
uint8_t, 114
include/arch/x86/ioports.h
inb, 60
inl, 60
outb, 60
outl, 60
include/arch/x86/types.h
int16_t, 115
int64_t, 115
int8_t, 115
intptr_t, 115
size_t, 115
uint16_t, 115
uint32_t, 115
uint64_t, 115
uint8_t, 115
include/types.h
bool_t, 116
FALSE, 116
NULL, 116
pok_blackboard_id_t, 116
pok_bool_t, 116
pok_buffer_id_t, 116
pok_event_id_t, 116
pok_lockobj_id_t, 117
pok_partition_id_t, 117
pok_port_direction_t, 117
pok_port_id_t, 117
pok_port_kind_t, 117
pok_port_size_t, 117
pok_queueing_discipline_t, 117
pok_range_t, 117
pok_sem_id_t, 117
pok_sem_value_t, 117
pok_size_t, 117
TRUE, 116
index
pok_port_t, 33
initial_value
pok_lockobj_attr_t, 25

initialized
pok_lockobj_t, 27
inl
include/arch/x86/ioports.h, 60
int16_t
arch/x86/types.h, 112
include/arch/sparc/types.h, 113
include/arch/x86/types.h, 115
int64_t
arch/x86/types.h, 112
include/arch/sparc/types.h, 113
include/arch/x86/types.h, 115
int8_t
arch/x86/types.h, 112
include/arch/sparc/types.h, 113
include/arch/x86/types.h, 115
interrupt.c
update_tss, 111
interrupt.h
INTERRUPT_HANDLER, 142
INTERRUPT_HANDLER_errorcode, 142
INTERRUPT_HANDLER_syscall, 142
pok_tss, 143
update_tss, 143
interrupt_frame, 22
__esp, 22
cs, 22
ds, 22
eax, 22
ebp, 22
ebx, 22
ecx, 23
edi, 23
edx, 23
eflags, 23
eip, 23
error, 23
es, 23
esi, 23
esp, 23
ss, 23
intptr_t
arch/x86/types.h, 112
include/arch/sparc/types.h, 113
include/arch/x86/types.h, 115
io_bit_map_offset
__attribute__, 10
irq.h
ack_irq, 85
IRQMP_BASE, 85
IRQMP_CLEAR_OFFSET, 85
IRQMP_MASK0_OFFSET, 85
unmask_irq, 85
is_locked
pok_lockobj_t, 27

KERNEL_STACK_SIZE
ppc/space.c, 62
sparc/space.c, 66
x86/space.c, 70
kernel.h
pok_kernel_restart, 149
pok_kernel_stop, 149
kernel_sp
space_context_t, 38
kind
pok_lockobj_attr_t, 25
pok_lockobj_t, 27
pok_port_t, 33

L0_OFFSET
context_offset.h, 83
L1_OFFSET
context_offset.h, 83
L2_OFFSET
context_offset.h, 83
L3_OFFSET
context_offset.h, 83
L4_OFFSET
context_offset.h, 83
L5_OFFSET
context_offset.h, 83
L6_OFFSET
context_offset.h, 83
L7_OFFSET
context_offset.h, 83
LOCKOBJ_LOCK_TIMED
lockobj.h, 152
LOCKOBJ_OPERATION_BROADCAST
lockobj.h, 152
LOCKOBJ_OPERATION_LOCK
lockobj.h, 152
LOCKOBJ_OPERATION_SIGNAL
lockobj.h, 152
LOCKOBJ_OPERATION_UNLOCK
lockobj.h, 152
LOCKOBJ_OPERATION_WAIT
lockobj.h, 152
LOCKOBJ_STATE_LOCK
lockobj.h, 152
LOCKOBJ_STATE_UNLOCK
lockobj.h, 152
LOCKOBJ_STATE_WAITEVENT
lockobj.h, 152
LOCKOBK_LOCK_REGULAR
lockobj.h, 152
LEON_CTX_NBR
sparc/space.h, 91
last_receive
pok_port_t, 34
ldt
__attribute__, 10
length_high
pok_memory_map_t, 28
length_low
pok_memory_map_t, 29
libc.h
memcpy, 164

memset, 164
strcmp, 164
strlen, 164
strncmp, 164
limit
__attribute__, 10
limit_high
__attribute__, 10
limit_low
__attribute__, 10
load_addr
pok_multiboot_header_t, 30
load_end_addr
pok_multiboot_header_t, 31
loader.h
pok_loader_load_partition, 150
lock
pok_port_t, 34
lock_kind
pok_lockobj_lockattr_t, 26
locking_policy
pok_lockobj_attr_t, 26
pok_lockobj_t, 27
lockobj.h
LOCKOBJ_LOCK_TIMED, 152
LOCKOBJ_OPERATION_BROADCAST, 152
LOCKOBJ_OPERATION_LOCK, 152
LOCKOBJ_OPERATION_SIGNAL, 152
LOCKOBJ_OPERATION_UNLOCK, 152
LOCKOBJ_OPERATION_WAIT, 152
LOCKOBJ_STATE_LOCK, 152
LOCKOBJ_STATE_UNLOCK, 152
LOCKOBJ_STATE_WAITEVENT, 152
LOCKOBK_LOCK_REGULAR, 152
POK_LOCKOBJ_KIND_EVENT, 151
POK_LOCKOBJ_KIND_MUTEX, 151
POK_LOCKOBJ_KIND_SEMAPHORE, 151
POK_LOCKOBJ_POLICY_PCP, 151
POK_LOCKOBJ_POLICY_PIP, 151
POK_LOCKOBJ_POLICY_STANDARD, 151
lockobj.h
pok_locking_policy_t, 151
pok_lockobj_create, 152
pok_lockobj_eventbroadcast, 152
pok_lockobj_eventsignal, 152
pok_lockobj_eventwait, 152
pok_lockobj_init, 153
pok_lockobj_kind_t, 151
pok_lockobj_lock, 153
pok_lockobj_lock_kind_t, 151
pok_lockobj_operation_t, 152
pok_lockobj_partition_create, 153
pok_lockobj_partition_wrapper, 153
pok_lockobj_unlock, 153
pok_mutex_state_t, 152
lr
context_t, 14
volatile_context_t, 40
MEM_16MB
pm.h, 126
MM_ACC_E
sparc/space.h, 91
MM_ACC_R
sparc/space.h, 91
MM_ACC_R_S_RW
sparc/space.h, 91
MM_ACC_RE
sparc/space.h, 91
MM_ACC_RW
sparc/space.h, 91
MM_ACC_RWE
sparc/space.h, 92
MM_ACC_S_RE
sparc/space.h, 92
MM_ACC_S_RWE
sparc/space.h, 92
MM_CACHEABLE
sparc/space.h, 92
MM_ET_INVALID
sparc/space.h, 92
MM_ET_PTD
sparc/space.h, 92
MM_ET_PTE
sparc/space.h, 92
MM_LVL1_ENTRIES_NBR
sparc/space.h, 93
MM_LVL1_PAGE_SIZE
sparc/space.h, 93
MM_LVL2_ENTRIES_NBR
sparc/space.h, 93
MM_LVL2_PAGE_SIZE
sparc/space.h, 93
MM_LVL3_ENTRIES_NBR
sparc/space.h, 93
MM_LVL3_PAGE_SIZE
sparc/space.h, 93
MM_MODIFIED
sparc/space.h, 93
MM_REFERENCED
sparc/space.h, 93
MMU_CTRL_REG
sparc/space.h, 93
MMU_CTX_REG
sparc/space.h, 94
MMU_CTXTBL_PTR
sparc/space.h, 94
MMU_FAULT_ADDR
sparc/space.h, 94
MMU_FAULT_STATUS
sparc/space.h, 94
MSR_DR
msr.h, 49
MSR_EE
msr.h, 49
MSR_IP
msr.h, 50

MSR_IR
msr.h, 50
MSR_PR
msr.h, 50
MULTIBOOT_CMDLINE
multiboot.h, 144
MULTIBOOT_MODS
multiboot.h, 144
magic
pok_multiboot_header_t, 31
max_value
pok_lockobj_attr_t, 26
pok_lockobj_t, 28
mem_lower
pok_multiboot_info_t, 32
mem_upper
pok_multiboot_info_t, 32
memcpy
libc.h, 164
memcpy.c, 167
memcpy.c
memcpy, 167
memset
libc.h, 164
memset.c
__attribute__, 168
mm_index1
sparc/space.h, 92
mm_index2
sparc/space.h, 92
mm_index3
sparc/space.h, 93
mmap_addr
pok_multiboot_info_t, 32
mmap_length
pok_multiboot_info_t, 32
mod_end
pok_module_t, 29
mod_start
pok_module_t, 29
mods_addr
pok_multiboot_info_t, 32
mods_count
pok_multiboot_info_t, 32
msr.h
MSR_DR, 49
MSR_EE, 49
MSR_IP, 50
MSR_IR, 50
MSR_PR, 50
multiboot.h
EXT_C, 144
MULTIBOOT_CMDLINE, 144
MULTIBOOT_MODS, 144
must_be_flushed
pok_port_t, 34

NPC_OFFSET
context_offset.h, 84
NULL
include/types.h, 116
nargs
pok_syscall_args_t, 36
next_header
cpio_file, 18
num
pok_elf_section_header_table_t, 25

OSCILLATOR_RATE
pit.c, 122
obj_kind
pok_lockobj_lockattr_t, 26
off_b
pok_port_t, 34
off_e
pok_port_t, 34
offset_high
__attribute__, 11
offset_low
__attribute__, 11
op_size
__attribute__, 11
operation
pok_lockobj_lockattr_t, 26
outb
arch/ppc/prep/ioports.h, 59
include/arch/x86/ioports.h, 60
outl
include/arch/x86/ioports.h, 60

POK_ERRNO_DIRECTION
errno.h, 162
POK_ERRNO_DISCIPLINE
errno.h, 163
POK_ERRNO_EDOM
errno.h, 162
POK_ERRNO_EFAULT
errno.h, 162
POK_ERRNO_EINVAL
errno.h, 162
POK_ERRNO_EMPTY
errno.h, 163
POK_ERRNO_EPERM
errno.h, 162
POK_ERRNO_ERANGE
errno.h, 162
POK_ERRNO_EXISTS
errno.h, 162
POK_ERRNO_FULL
errno.h, 163
POK_ERRNO_HUGE_VAL
errno.h, 162
POK_ERRNO_KIND
errno.h, 163
POK_ERRNO_LOCKOBJ_KIND
errno.h, 163
POK_ERRNO_LOCKOBJ_NOTREADY
errno.h, 163

POK_ERRNO_LOCKOBJ_POLICY
errno.h, 163
POK_ERRNO_LOCKOBJ_UNAVAILABLE
errno.h, 163
POK_ERRNO_MODE
errno.h, 163
POK_ERRNO_NOTFOUND
errno.h, 162
POK_ERRNO_OK
errno.h, 162
POK_ERRNO_PARAM
errno.h, 162
POK_ERRNO_PARTITION
errno.h, 163
POK_ERRNO_PARTITION_ATTR
errno.h, 162
POK_ERRNO_PARTITION_MODE
errno.h, 163
POK_ERRNO_PORT
errno.h, 162
POK_ERRNO_PORTPART
errno.h, 163
POK_ERRNO_READY
errno.h, 163
POK_ERRNO_SIZE
errno.h, 162
POK_ERRNO_THREAD
errno.h, 162
POK_ERRNO_THREADATTR
errno.h, 162
POK_ERRNO_TIME
errno.h, 162
POK_ERRNO_TIMEOUT
errno.h, 163
POK_ERRNO_TOOMANY
errno.h, 162
POK_ERRNO_UNAVAILABLE
errno.h, 162
POK_LOCKOBJ_KIND_EVENT
lockobj.h, 151
POK_LOCKOBJ_KIND_MUTEX
lockobj.h, 151
POK_LOCKOBJ_KIND_SEMAPHORE
lockobj.h, 151
POK_LOCKOBJ_POLICY_PCP
lockobj.h, 151
POK_LOCKOBJ_POLICY_PIP
lockobj.h, 151
POK_LOCKOBJ_POLICY_STANDARD
lockobj.h, 151
POK_PORT_DIRECTION_IN
port.h, 165
POK_PORT_DIRECTION_OUT
port.h, 165
POK_PORT_KIND_INVALID
port.h, 166
POK_PORT_KIND_QUEUEING
port.h, 166
POK_PORT_KIND_SAMPLING
port.h, 166
POK_PORT_QUEUEING_DISCIPLINE_FIFO
port.h, 166
POK_PORT_QUEUEING_DISCIPLINE_PRIORITY
port.h, 166
POK_SCHED_EDF
schedvalues.h, 153
POK_SCHED_FIFO
schedvalues.h, 153
POK_SCHED_GLOBAL_TIMESLICE
schedvalues.h, 153
POK_SCHED_LLF
schedvalues.h, 153
POK_SCHED_RMS
schedvalues.h, 153
POK_SCHED_RR
schedvalues.h, 153
POK_SCHED_STATIC
schedvalues.h, 153
POK_SYSCALL_CONSWRITE
syscall.h, 155
POK_SYSCALL_GETTICK
syscall.h, 155
POK_SYSCALL_INT_NUMBER
syscall.h, 155
POK_SYSCALL_THREAD_CREATE
syscall.h, 155
POK_SYSCALL_THREAD_DELAYED_START
syscall.h, 155
POK_SYSCALL_THREAD_ID
syscall.h, 155
POK_SYSCALL_THREAD_PERIOD
syscall.h, 155
POK_SYSCALL_THREAD_RESTART
syscall.h, 155
POK_SYSCALL_THREAD_RESUME
syscall.h, 155
POK_SYSCALL_THREAD_SET_PRIORITY
syscall.h, 155
POK_SYSCALL_THREAD_SLEEP
syscall.h, 155
POK_SYSCALL_THREAD_SLEEP_UNTIL
syscall.h, 155
POK_SYSCALL_THREAD_STATUS
syscall.h, 155
POK_SYSCALL_THREAD_STOP
syscall.h, 155
POK_SYSCALL_THREAD_STOPSELF
syscall.h, 155
POK_SYSCALL_THREAD_SUSPEND
syscall.h, 155
POK_SYSCALL_THREAD_SUSPEND_TARGET
syscall.h, 155
p_align
Elf32_Phdr, 21
p_filesz
Elf32_Phdr, 21

p_flags
Elf32_Phdr, 21
p_memsz
Elf32_Phdr, 21
p_offset
Elf32_Phdr, 21
p_paddr
Elf32_Phdr, 21
p_type
Elf32_Phdr, 21
p_vaddr
Elf32_Phdr, 21
PARTITION_ID
x86/syscalls.c, 75
PC_OFFSET
context_offset.h, 84
PIC_MASTER_BASE
pic.h, 120
PIC_MASTER_ICW1
pic.h, 120
PIC_MASTER_ICW2
pic.h, 120
PIC_MASTER_ICW3
pic.h, 120
PIC_MASTER_ICW4
pic.h, 120
PIC_SLAVE_BASE
pic.h, 120
PIC_SLAVE_ICW1
pic.h, 120
PIC_SLAVE_ICW2
pic.h, 120
PIC_SLAVE_ICW3
pic.h, 120
PIC_SLAVE_ICW4
pic.h, 120
PIT_BASE
pit.c, 122
PIT_IRQ
pit.c, 122
POK_FATAL
debug.h, 149
POK_PAGE_MASK
ppc/space.c, 62
POK_PAGE_SIZE
ppc/space.c, 62
POK_PORT_MAX_SIZE
port.h, 165
POK_PREP_IOBASE
arch/ppc/prep/ioports.h, 59
PPC_PTE_C
ppc/space.c, 62
PPC_PTE_G
ppc/space.c, 62
PPC_PTE_H
ppc/space.c, 62
PPC_PTE_I
ppc/space.c, 62
PPC_PTE_M
ppc/space.c, 62
PPC_PTE_PP_NO
ppc/space.c, 62
PPC_PTE_PP_RO
ppc/space.c, 62
PPC_PTE_PP_RW
ppc/space.c, 62
PPC_PTE_R
ppc/space.c, 62
PPC_PTE_V
ppc/space.c, 62
PPC_PTE_W
ppc/space.c, 63
PPC_SR_KP
ppc/space.c, 63
PPC_SR_Ks
ppc/space.c, 63
PPC_SR_T
ppc/space.c, 63
PSR_CWP_MASK
psr.h, 89
PSR_ET
psr.h, 89
PSR_OFFSET
context_offset.h, 84
PSR_PIL
psr.h, 89
PSR_PS
psr.h, 89
PSR_S
psr.h, 89
pad
context_t, 14
pad0
volatile_context_t, 40
pad1
volatile_context_t, 40
padding
__attribute__, 11
partition
pok_port_t, 34
pok_syscall_info_t, 36
partition_id
space_context_t, 38
phys_base
pok_space, 35
pic.c
pok_pic_eoi, 118
pok_pic_init, 118
pok_pic_mask, 118
pok_pic_unmask, 119
pic.h
PIC_MASTER_BASE, 120
PIC_MASTER_ICW1, 120
PIC_MASTER_ICW2, 120
PIC_MASTER_ICW3, 120
PIC_MASTER_ICW4, 120

PIC_SLAVE_BASE, 120
PIC_SLAVE_ICW1, 120
PIC_SLAVE_ICW2, 120
PIC_SLAVE_ICW3, 120
PIC_SLAVE_ICW4, 120
pok_pic_eoi, 121
pok_pic_init, 121
pok_pic_mask, 121
pok_pic_unmask, 121
pit.c
INTERRUPT_HANDLER, 123
OSCILLATOR_RATE, 122
PIT_BASE, 122
PIT_IRQ, 122
pok_x86_qemu_timer_init, 123
pit.h
pok_x86_qemu_timer_init, 123
pm.c
__pok_begin, 125
__pok_end, 125
ALIGN_UP, 124
pok_multiboot_info, 125
pok_multiboot_magic, 125
pok_pm_init, 124
pok_pm_sbrk, 125
pok_x86_pm_brk, 125
pok_x86_pm_heap_end, 125
pok_x86_pm_heap_start, 125
pm.h
MEM_16MB, 126
pok_pm_init, 126
pok_pm_sbrk, 126
pok_aout_symbol_table_t, 24
addr, 24
reserved, 24
strsize, 24
tabsize, 24
pok_arch_decr_int
ppc/timer.c, 79
pok_arch_dsi_int
ppc/space.c, 63
pok_arch_event_register
arch.h, 135
ppc/arch.c, 43
sparc/arch.c, 46
x86/arch.c, 48
pok_arch_idle
arch.h, 135
ppc/arch.c, 44
sparc/arch.c, 46
x86/arch.c, 48
pok_arch_init
arch.h, 135
ppc/arch.c, 44
sparc/arch.c, 46
x86/arch.c, 48
pok_arch_isi_int
ppc/space.c, 63
pok_arch_preempt_disable
arch.h, 136
ppc/arch.c, 44
sparc/arch.c, 46
x86/arch.c, 48
pok_arch_preempt_enable
arch.h, 136
ppc/arch.c, 44
sparc/arch.c, 47
x86/arch.c, 49
pok_arch_rfi
ppc/space.c, 64
pok_arch_sc_int
ppc/syscalls.c, 72
sparc/syscalls.c, 73
pok_arch_sp
arch/sparc/thread.h, 78
pok_arch_space_init
ppc/arch.c, 44
ppc/space.c, 64
sparc/space.c, 67
sparc/space.h, 94
pok_blackboard_id_t
include/types.h, 116
pok_bool_t
include/types.h, 116
pok_boot
boot.c, 127
boot.h, 147
pok_bsp_init
bsp.h, 145
ppc/prep/bsp.c, 50
sparc/leon3/bsp.c, 51
x86/x86-qemu/bsp.c, 52
pok_bsp_irq_acknowledge
bsp.h, 145
x86/x86-qemu/bsp.c, 52
pok_bsp_irq_register
bsp.h, 146
x86/x86-qemu/bsp.c, 52
pok_bsp_mem_alloc
bsp.h, 146
ppc/prep/bsp.c, 50
sparc/leon3/bsp.c, 51
x86/x86-qemu/bsp.c, 53
pok_bsp_time_init
bsp.h, 146
ppc/timer.c, 79
sparc/leon3/timer.c, 80
x86/x86-qemu/bsp.c, 53
pok_buffer_id_t
include/types.h, 116
pok_cons_init
arch/ppc/prep/cons.c, 53
arch/ppc/prep/cons.h, 55
arch/sparc/leon3/cons.c, 54
arch/sparc/leon3/cons.h, 58
arch/x86/x86-qemu/cons.c, 54

arch/x86/x86-qemu/cons.h, 58
pok_cons_write
bsp.h, 146
pok_context_create
arch.h, 136
arch/ppc/thread.h, 77
arch/sparc/thread.h, 78
arch/x86/thread.h, 78
pok_context_reset
arch.h, 136
arch/x86/thread.h, 78
pok_context_switch
arch.h, 136
arch/ppc/thread.h, 77
arch/sparc/thread.h, 78
arch/x86/thread.h, 78
pok_core_syscall
syscall.c, 129
syscall.h, 156
pok_create_space
arch.h, 136
ppc/space.c, 64
sparc/space.c, 68
x86/space.c, 70
pok_dispatch_space
arch.h, 137
x86/space.c, 70
pok_elf_section_header_table_t, 24
addr, 25
num, 25
shndx, 25
size, 25
pok_event_id_t
include/types.h, 116
pok_event_init
event.c, 99
event.h, 103
pok_exception_init
event.h, 103
pok_gdt
gdt.c, 107
pok_gdt_init
gdt.c, 106
gdt.h, 110
pok_idt
event.c, 100
pok_idt_init
event.c, 99
event.h, 103
pok_idt_set_gate
event.c, 99
event.h, 103
pok_kernel_restart
kernel.h, 149
pok_kernel_stop
kernel.h, 149
pok_loader_load_partition
loader.h, 150
pok_locking_policy_t
lockobj.h, 151
pok_lockobj_attr_t, 25
initial_value, 25
kind, 25
locking_policy, 26
max_value, 26
queueing_policy, 26
pok_lockobj_create
lockobj.h, 152
pok_lockobj_eventbroadcast
lockobj.h, 152
pok_lockobj_eventsignal
lockobj.h, 152
pok_lockobj_eventwait
lockobj.h, 152
pok_lockobj_id_t
include/types.h, 117
pok_lockobj_init
lockobj.h, 153
pok_lockobj_kind_t
lockobj.h, 151
pok_lockobj_lock
lockobj.h, 153
pok_lockobj_lock_kind_t
lockobj.h, 151
pok_lockobj_lockattr_t, 26
lock_kind, 26
obj_kind, 26
operation, 26
time, 26
pok_lockobj_operation_t
lockobj.h, 152
pok_lockobj_partition_create
lockobj.h, 153
pok_lockobj_partition_wrapper
lockobj.h, 153
pok_lockobj_t, 27
current_value, 27
eventspin, 27
initialized, 27
is_locked, 27
kind, 27
locking_policy, 27
max_value, 28
queueing_policy, 28
spin, 28
thread_state, 28
pok_lockobj_unlock
lockobj.h, 153
pok_memory_map_t, 28
base_addr_high, 28
base_addr_low, 28
length_high, 28
length_low, 29
size, 29
type, 29
pok_module_t, 29

mod_end, 29
mod_start, 29
reserved, 29
string, 29
pok_multiboot_header_t, 30
bss_end_addr, 30
checksum, 30
entry_addr, 30
flags, 30
header_addr, 30
load_addr, 30
load_end_addr, 31
magic, 31
pok_multiboot_info
pm.c, 125
pok_multiboot_info_t, 31
aout_sym, 31
boot_device, 31
cmdline, 31
elf_sec, 32
flags, 32
mem_lower, 32
mem_upper, 32
mmap_addr, 32
mmap_length, 32
mods_addr, 32
mods_count, 32
u, 32
pok_multiboot_magic
pm.c, 125
pok_mutex_state_t
lockobj.h, 152
pok_partition_id_t
include/types.h, 117
pok_pic_eoi
pic.c, 118
pic.h, 121
pok_pic_init
pic.c, 118
pic.h, 121
pok_pic_mask
pic.c, 118
pic.h, 121
pok_pic_unmask
pic.c, 119
pic.h, 121
pok_pm_init
pm.c, 124
pm.h, 126
pok_pm_sbrk
pm.c, 125
pm.h, 126
pok_port_direction_t
include/types.h, 117
pok_port_directions_t
port.h, 165
pok_port_id_t
include/types.h, 117
pok_port_kind_t
include/types.h, 117
pok_port_kinds_t
port.h, 166
pok_port_queueing_discipline_t
port.h, 165
pok_port_queueing_disciplines_t
port.h, 166
pok_port_size_t
include/types.h, 117
pok_port_t, 32
direction, 33
discipline, 33
empty, 33
full, 33
identifier, 33
index, 33
kind, 33
last_receive, 34
lock, 34
must_be_flushed, 34
off_b, 34
off_e, 34
partition, 34
ready, 34
refresh, 34
size, 34
pok_queueing_discipline_t
include/types.h, 117
pok_range_t
include/types.h, 117
pok_ret_t
errno.h, 162
pok_sched_t
schedvalues.h, 153
pok_sem_id_t
include/types.h, 117
pok_sem_value_t
include/types.h, 117
pok_size_t
include/types.h, 117
pok_space, 34
phys_base, 35
size, 35
pok_space_base_vaddr
arch.h, 137
ppc/space.c, 64
sparc/space.c, 68
x86/space.c, 71
pok_space_context_create
arch.h, 137
ppc/space.c, 64
sparc/space.c, 68
x86/space.c, 71
pok_space_context_restart
arch.h, 138
pok_space_switch
arch.h, 138

ppc/space.c, 65
sparc/space.c, 69
x86/space.c, 72
pok_sparc_isr
traps.c, 97
traps.h, 98
pok_spinlock_t
ppc/spinlock.h, 139
sparc/spinlock.h, 140
x86/spinlock.h, 141
pok_syscall_args_t, 35
arg1, 35
arg2, 35
arg3, 35
arg4, 36
arg5, 36
nargs, 36
pok_syscall_id_t
syscall.h, 155
pok_syscall_info_t, 36
base_addr, 36
partition, 36
thread, 36
pok_syscall_init
event.h, 104
syscall.h, 160
x86/syscalls.c, 75
pok_syscalls_init
sparc/syscalls.c, 74
syscalls.h, 95
pok_thread_stack_addr
arch.h, 138
ppc/arch.c, 45
sparc/arch.c, 47
x86/arch.c, 49
pok_tss
gdt.c, 107
interrupt.h, 143
pok_tss_init
gdt.c, 107
gdt.h, 110
pok_x86_pm_brk
pm.c, 125
pok_x86_pm_heap_end
pm.c, 125
pok_x86_pm_heap_start
pm.c, 125
pok_x86_qemu_timer_init
pit.c, 123
pit.h, 123
port.h
POK_PORT_DIRECTION_IN, 165
POK_PORT_DIRECTION_OUT, 165
POK_PORT_KIND_INVALID, 166
POK_PORT_KIND_QUEUEING, 166
POK_PORT_KIND_SAMPLING, 166
POK_PORT_QUEUEING_DISCIPLINE_FIFO, 166
POK_PORT_QUEUEING_DISCIPLINE_PRIORITY, 166
port.h
POK_PORT_MAX_SIZE, 165
pok_port_directions_t, 165
pok_port_kinds_t, 166
pok_port_queueing_discipline_t, 165
pok_port_queueing_disciplines_t, 166
ppc/arch.c
pok_arch_event_register, 43
pok_arch_idle, 44
pok_arch_init, 44
pok_arch_preempt_disable, 44
pok_arch_preempt_enable, 44
pok_arch_space_init, 44
pok_thread_stack_addr, 45
ppc/prep/bsp.c
_end, 51
pok_bsp_init, 50
pok_bsp_mem_alloc, 50
ppc/space.c
KERNEL_STACK_SIZE, 62
POK_PAGE_MASK, 62
POK_PAGE_SIZE, 62
PPC_PTE_C, 62
PPC_PTE_G, 62
PPC_PTE_H, 62
PPC_PTE_I, 62
PPC_PTE_M, 62
PPC_PTE_PP_NO, 62
PPC_PTE_PP_RO, 62
PPC_PTE_PP_RW, 62
PPC_PTE_R, 62
PPC_PTE_V, 62
PPC_PTE_W, 63
PPC_SR_KP, 63
PPC_SR_Ks, 63
PPC_SR_T, 63
pok_arch_dsi_int, 63
pok_arch_isi_int, 63
pok_arch_rfi, 64
pok_arch_space_init, 64
pok_create_space, 64
pok_space_base_vaddr, 64
pok_space_context_create, 64
pok_space_switch, 65
spaces, 65
ppc/spinlock.h
pok_spinlock_t, 139
SPIN_LOCK, 139
SPIN_UNLOCK, 139
ppc/syscalls.c
pok_arch_sc_int, 72
ppc/timer.c
BUS_FREQ, 79
FREQ_DIV, 79
pok_arch_decr_int, 79
pok_bsp_time_init, 79

ppc_pte_t, 37
    rpn_flags, 37
    vsid_api, 37
present
    __attribute__, 11
psr.h
    PSR_CWP_MASK, 89
    PSR_ET, 89
    PSR_PIL, 89
    PSR_PS, 89
    PSR_S, 89
ptd
    sparc/space.h, 94
pte
    sparc/space.h, 94

queueing_policy
    pok_lockobj_attr_t, 26
    pok_lockobj_t, 28

r0
    volatile_context_t, 40
r10
    volatile_context_t, 40
r11
    volatile_context_t, 40
r12
    volatile_context_t, 40
r13
    context_t, 14
    volatile_context_t, 40
r14
    context_t, 14
r15
    context_t, 14
r16
    context_t, 14
r17
    context_t, 14
r18
    context_t, 15
r19
    context_t, 15
r2
    context_t, 15
    volatile_context_t, 40
r20
    context_t, 15
r21
    context_t, 15
r22
    context_t, 15
r23
    context_t, 15
r24
    context_t, 15
r25
    context_t, 15
r26
    context_t, 15
r27
    context_t, 15
r28
    context_t, 15
r29
    context_t, 16
r3
    volatile_context_t, 41
r30
    context_t, 16
r31
    context_t, 16
r4
    volatile_context_t, 41
r5
    volatile_context_t, 41
r6
    volatile_context_t, 41
r7
    volatile_context_t, 41
r8
    volatile_context_t, 41
r9
    volatile_context_t, 41
RESTORE_CNT_OFFSET
    context_offset.h, 84
ready
    pok_port_t, 34
refresh
    pok_port_t, 34
res0
    __attribute__, 11
res1
    __attribute__, 11
reserved
    pok_aout_symbol_table_t, 24
    pok_module_t, 29
rpn_flags
    ppc_pte_t, 37

s
    __attribute__, 11
SPARC_PAGE_SIZE
    sparc_conf.h, 86
SPARC_PROC_FREQ
    sparc_conf.h, 86
SPARC_RAM_ADDR
    sparc_conf.h, 86
SPARC_TRAP_IRQ_BASE
    traps.h, 98
SPIN_LOCK
    ppc/spinlock.h, 139
    sparc/spinlock.h, 140
    x86/spinlock.h, 141
SPIN_UNLOCK
    ppc/spinlock.h, 139
    sparc/spinlock.h, 140
    x86/spinlock.h, 141

schedvalues.h
POK_SCHED_EDF, 153
POK_SCHED_FIFO, 153
POK_SCHED_GLOBAL_TIMESLICE, 153
POK_SCHED_LLF, 153
POK_SCHED_RMS, 153
POK_SCHED_RR, 153
POK_SCHED_STATIC, 153
schedvalues.h
pok_sched_t, 153
segsel
__attribute__, 11
shndx
pok_elf_section_header_table_t, 25
size
pok_elf_section_header_table_t, 25
pok_memory_map_t, 29
pok_port_t, 34
pok_space, 35
size_t
arch/x86/types.h, 112
include/arch/sparc/types.h, 114
include/arch/x86/types.h, 115
sp
context_t, 16
volatile_context_t, 41
space_context_t, 37
arg1, 37
arg2, 37
ctx, 38
fake_ret, 38
kernel_sp, 38
partition_id, 38
user_pc, 38
user_sp, 38
spaces
ppc/space.c, 65
sparc/space.c, 69
sparc/arch.c
pok_arch_event_register, 46
pok_arch_idle, 46
pok_arch_init, 46
pok_arch_preempt_disable, 46
pok_arch_preempt_enable, 47
pok_thread_stack_addr, 47
sparc/leon3/bsp.c
_end, 52
pok_bsp_init, 51
pok_bsp_mem_alloc, 51
sparc/leon3/timer.c
pok_bsp_time_init, 80
timer_isr, 81
sparc/space.c
__attribute__, 67
KERNEL_STACK_SIZE, 66
pok_arch_space_init, 67
pok_create_space, 68
pok_space_base_vaddr, 68
pok_space_context_create, 68
pok_space_switch, 69
spaces, 69
sparc/space.h
ASI_M_MMUREGS, 91
LEON_CTX_NBR, 91
MM_ACC_E, 91
MM_ACC_R, 91
MM_ACC_R_S_RW, 91
MM_ACC_RE, 91
MM_ACC_RW, 91
MM_ACC_RWE, 92
MM_ACC_S_RE, 92
MM_ACC_S_RWE, 92
MM_CACHEABLE, 92
MM_ET_INVALID, 92
MM_ET_PTD, 92
MM_ET_PTE, 92
MM_LVL1_ENTRIES_NBR, 93
MM_LVL1_PAGE_SIZE, 93
MM_LVL2_ENTRIES_NBR, 93
MM_LVL2_PAGE_SIZE, 93
MM_LVL3_ENTRIES_NBR, 93
MM_LVL3_PAGE_SIZE, 93
MM_MODIFIED, 93
MM_REFERENCED, 93
MMU_CTRL_REG, 93
MMU_CTX_REG, 94
MMU_CTXTBL_PTR, 94
MMU_FAULT_ADDR, 94
MMU_FAULT_STATUS, 94
mm_index1, 92
mm_index2, 92
mm_index3, 93
pok_arch_space_init, 94
ptd, 94
pte, 94
sparc/spinlock.h
pok_spinlock_t, 140
SPIN_LOCK, 140
SPIN_UNLOCK, 140
sparc/syscalls.c
pok_arch_sc_int, 73
pok_syscalls_init, 74
sparc_conf.h
ASI_MMU_BYPASS, 86
SPARC_PAGE_SIZE, 86
SPARC_PROC_FREQ, 86
SPARC_RAM_ADDR, 86
WINDOWS_NBR, 86
sparc_traps_handler
traps.h, 98
spin
pok_lockobj_t, 28
srr0
volatile_context_t, 41
srr1
volatile_context_t, 41

ss
__attribute__, 11
interrupt_frame, 23
ss0
__attribute__, 11
ss1
__attribute__, 11
ss2
__attribute__, 12
start_context_t, 38
ctx, 39
entry, 39
fake_ret, 39
id, 39
strcmp
libc.h, 164
string
pok_module_t, 29
strlen
libc.h, 164
strncmp
libc.h, 164
strsize
pok_aout_symbol_table_t, 24
syscall.h
POK_SYSCALL_CONSWRITE, 155
POK_SYSCALL_GETTICK, 155
POK_SYSCALL_INT_NUMBER, 155
POK_SYSCALL_THREAD_CREATE, 155
POK_SYSCALL_THREAD_DELAYED_START, 155
POK_SYSCALL_THREAD_ID, 155
POK_SYSCALL_THREAD_PERIOD, 155
POK_SYSCALL_THREAD_RESTART, 155
POK_SYSCALL_THREAD_RESUME, 155
POK_SYSCALL_THREAD_SET_PRIORITY, 155
POK_SYSCALL_THREAD_SLEEP, 155
POK_SYSCALL_THREAD_SLEEP_UNTIL, 155
POK_SYSCALL_THREAD_STATUS, 155
POK_SYSCALL_THREAD_STOP, 155
POK_SYSCALL_THREAD_STOPSELF, 155
POK_SYSCALL_THREAD_SUSPEND, 155
POK_SYSCALL_THREAD_SUSPEND_TARGET, 155
syscall.c
pok_core_syscall, 129
syscall.h
pok_core_syscall, 156
pok_syscall_id_t, 155
pok_syscall_init, 160
syscalls.h
pok_syscalls_init, 95

TIMER1
timer.h, 87
TIMER_CTRL_CH
timer.h, 87
TIMER_CTRL_DH
timer.h, 87
TIMER_CTRL_EN
timer.h, 87
TIMER_CTRL_IE
timer.h, 88
TIMER_CTRL_IP
timer.h, 88
TIMER_CTRL_LD
timer.h, 88
TIMER_CTRL_OFFSET
timer.h, 88
TIMER_CTRL_RS
timer.h, 88
TIMER_IRQ
timer.h, 88
TIMER_RELOAD_OFFSET
timer.h, 88
TIMER_SCALER_OFFSET
timer.h, 88
TRUE
include/types.h, 116
tabsize
pok_aout_symbol_table_t, 24
thread
pok_syscall_info_t, 36
thread_state
pok_lockobj_t, 28
time
pok_lockobj_lockattr_t, 26
timer.h
TIMER1, 87
TIMER_CTRL_CH, 87
TIMER_CTRL_DH, 87
TIMER_CTRL_EN, 87
TIMER_CTRL_IE, 88
TIMER_CTRL_IP, 88
TIMER_CTRL_LD, 88
TIMER_CTRL_OFFSET, 88
TIMER_CTRL_RS, 88
TIMER_IRQ, 88
TIMER_RELOAD_OFFSET, 88
TIMER_SCALER_OFFSET, 88
timer_isr
sparc/leon3/timer.c, 81
trace_trap
__attribute__, 12
trap_handler
traps.c, 96
traps.c
pok_sparc_isr, 97
trap_handler, 96
traps_init, 97
traps.h
pok_sparc_isr, 98
sparc_traps_handler, 98
traps_init, 98
traps_init
traps.c, 97
traps.h, 98

tss_set_esp0
gdt.c, 107
gdt.h, 111
type
__attribute__, 12
pok_memory_map_t, 29

u
pok_multiboot_info_t, 32
UART1
arch/sparc/leon3/cons.h, 56
UART_CTRL_FL
arch/sparc/leon3/cons.h, 56
UART_CTRL_LB
arch/sparc/leon3/cons.h, 56
UART_CTRL_OFFSET
arch/sparc/leon3/cons.h, 56
UART_CTRL_PE
arch/sparc/leon3/cons.h, 56
UART_CTRL_PS
arch/sparc/leon3/cons.h, 56
UART_CTRL_RE
arch/sparc/leon3/cons.h, 56
UART_CTRL_RI
arch/sparc/leon3/cons.h, 56
UART_CTRL_TE
arch/sparc/leon3/cons.h, 57
UART_CTRL_TI
arch/sparc/leon3/cons.h, 57
UART_DATA_OFFSET
arch/sparc/leon3/cons.h, 57
UART_SCALER_OFFSET
arch/sparc/leon3/cons.h, 57
UART_STAT_OFFSET
arch/sparc/leon3/cons.h, 57
UART_STATUS_BR
arch/sparc/leon3/cons.h, 57
UART_STATUS_DR
arch/sparc/leon3/cons.h, 57
UART_STATUS_ERR
arch/sparc/leon3/cons.h, 57
UART_STATUS_FE
arch/sparc/leon3/cons.h, 57
UART_STATUS_OE
arch/sparc/leon3/cons.h, 58
UART_STATUS_PE
arch/sparc/leon3/cons.h, 58
UART_STATUS_THE
arch/sparc/leon3/cons.h, 58
UART_STATUS_TSE
arch/sparc/leon3/cons.h, 58
uint16_t
arch/x86/types.h, 113
include/arch/sparc/types.h, 114
include/arch/x86/types.h, 115
uint32_t
arch/x86/types.h, 113
include/arch/sparc/types.h, 114
include/arch/x86/types.h, 115
uint64_t
arch/x86/types.h, 113
include/arch/sparc/types.h, 114
include/arch/x86/types.h, 115
uint8_t
arch/x86/types.h, 113
include/arch/sparc/types.h, 114
include/arch/x86/types.h, 115
unmask_irq
irq.h, 85
unused_lr
context_t, 16
volatile_context_t, 41
update_tss
interrupt.c, 111
interrupt.h, 143
user_pc
space_context_t, 38
user_sp
space_context_t, 38

volatile_context_t, 39
back_chain, 40
cr, 40
ctr, 40
lr, 40
pad0, 40
pad1, 40
r0, 40
r10, 40
r11, 40
r12, 40
r13, 40
r2, 40
r3, 41
r4, 41
r5, 41
r6, 41
r7, 41
r8, 41
r9, 41
sp, 41
srr0, 41
srr1, 41
unused_lr, 41
xer, 41
vsid_api
ppc_pte_t, 37

WIM_OFFSET
context_offset.h, 84
WINDOWS_NBR
sparc_conf.h, 86

x86/arch.c
pok_arch_event_register, 48
pok_arch_idle, 48
pok_arch_init, 48
pok_arch_preempt_disable, 48

pok_arch_preempt_enable, 49
pok_thread_stack_addr, 49
x86/space.c
KERNEL_STACK_SIZE, 70
pok_create_space, 70
pok_dispatch_space, 70
pok_space_base_vaddr, 71
pok_space_context_create, 71
pok_space_switch, 72
x86/spinlock.h
pok_spinlock_t, 141
SPIN_LOCK, 141
SPIN_UNLOCK, 141
x86/syscalls.c
INTERRUPT_HANDLER_syscall, 75
PARTITION_ID, 75
pok_syscall_init, 75
x86/x86-qemu/bsp.c
pok_bsp_init, 52
pok_bsp_irq_acknowledge, 52
pok_bsp_irq_register, 52
pok_bsp_mem_alloc, 53
pok_bsp_time_init, 53
xer
volatile_context_t, 41

Y_OFFSET
context_offset.h, 84