MPX

Generated by Doxygen 1.9.0

1 Data Structure Index 1

1.1 Data Structures . . . 1

2 File Index 3

2.1 File List . . . 3

3 Data Structure Documentation 5

3.1 alarm_s Struct Reference . . . 5

3.1.1 Field Documentation . . . 5

3.1.1.1 message . . . 6

3.1.1.2 next . . . 6

3.1.1.3 prev . . . 6

3.1.1.4 time . . . 6

3.2 aList Struct Reference . . . 6

3.2.1 Field Documentation . . . 7

3.2.1.1 count . . . 7

3.2.1.2 head . . . 7

3.2.1.3 tail . . . 7

3.3 CMCB_s Struct Reference . . . 7

3.3.1 Field Documentation . . . 8

3.3.1.1 namePCB . . . 8

3.3.1.2 next . . . 8

3.3.1.3 prev . . . 8

3.3.1.4 size . . . 8

3.3.1.5 start . . . 8

3.3.1.6 type . . . 9

3.4 Context Struct Reference . . . 9

3.4.1 Field Documentation . . . 9

3.4.1.1 cs . . . 9

3.4.1.2 ds . . . 9

3.4.1.3 eax . . . 10

3.4.1.4 ebp . . . 10

3.4.1.5 ebx . . . 10

3.4.1.6 ecx . . . 10

3.4.1.7 edi . . . 10

3.4.1.8 edx . . . 10

3.4.1.9 eflags . . . 10

3.4.1.10 eip . . . 10

3.4.1.11 es . . . 11

3.4.1.12 esi . . . 11

3.4.1.13 esp . . . 11

3.4.1.14 fs . . . 11

3.4.1.15 gs . . . 11

3.5 date_time Struct Reference . . . 11
3.5.1 Field Documentation . . . 12
3.5.1.1 day_m . . . 12
3.5.1.2 day_w . . . 12
3.5.1.3 day_y . . . 12
3.5.1.4 hour . . . 12
3.5.1.5 min . . . 12
3.5.1.6 mon . . . 12
3.5.1.7 sec . . . 12
3.5.1.8 year . . . 13
3.6 dcb_s Struct Reference . . . 13
3.6.1 Detailed Description . . . 13
3.6.2 Field Documentation . . . 13
3.6.2.1 buffer_loc . . . 14
3.6.2.2 buffer_ptr . . . 14
3.6.2.3 byte_count . . . 14
3.6.2.4 com_port . . . 14
3.6.2.5 count_ptr . . . 14
3.6.2.6 e_flag . . . 14
3.6.2.7 port_open . . . 14
3.6.2.8 status . . . 15
3.7 footer Struct Reference . . . 15
3.7.1 Field Documentation . . . 15
3.7.1.1 head . . . 15
3.8 gdt_descriptor_struct Struct Reference . . . 16
3.8.1 Field Documentation . . . 16
3.8.1.1 base . . . 16
3.8.1.2 limit . . . 16
3.9 gdt_entry_struct Struct Reference . . . 16
3.9.1 Field Documentation . . . 16
3.9.1.1 access . . . 17
3.9.1.2 base_high . . . 17
3.9.1.3 base_low . . . 17
3.9.1.4 base_mid . . . 17
3.9.1.5 flags . . . 17
3.9.1.6 limit_low . . . 17
3.10 header Struct Reference . . . 17
3.10.1 Field Documentation . . . 18
3.10.1.1 index_id . . . 18
3.10.1.2 size . . . 18
3.11 heap Struct Reference . . . 18
3.11.1 Field Documentation . . . 19

3.11.1.1 base . . . 19
3.11.1.2 index . . . 19
3.11.1.3 max_size . . . 19
3.11.1.4 min_size . . . 19
3.12 idt_entry_struct Struct Reference . . . 19
3.12.1 Field Documentation . . . 20
3.12.1.1 base_high . . . 20
3.12.1.2 base_low . . . 20
3.12.1.3 flags . . . 20
3.12.1.4 sselect . . . 20
3.12.1.5 zero . . . 20
3.13 idt_struct Struct Reference . . . 21
3.13.1 Field Documentation . . . 21
3.13.1.1 base . . . 21
3.13.1.2 limit . . . 21
3.14 index_entry Struct Reference . . . 21
3.14.1 Field Documentation . . . 21
3.14.1.1 block . . . 22
3.14.1.2 empty . . . 22
3.14.1.3 size . . . 22
3.15 index_table Struct Reference . . . 22
3.15.1 Field Documentation . . . 23
3.15.1.1 id . . . 23
3.15.1.2 table . . . 23
3.16 iod_s Struct Reference . . . 23
3.16.1 Field Documentation . . . 24
3.16.1.1 buffer_ptr . . . 24
3.16.1.2 count_ptr . . . 24
3.16.1.3 name . . . 24
3.16.1.4 next . . . 24
3.16.1.5 op_code . . . 24
3.16.1.6 pcb . . . 24
3.17 ioqueue_s Struct Reference . . . 25
3.17.1 Field Documentation . . . 25
3.17.1.1 count . . . 25
3.17.1.2 head . . . 25
3.17.1.3 tail . . . 26
3.18 LMCB_s Struct Reference . . . 26
3.18.1 Field Documentation . . . 26
3.18.1.1 size . . . 26
3.18.1.2 type . . . 26
3.19 mList Struct Reference . . . 27

3.19.1 Field Documentation . . . 27
3.19.1.1 count . . . 27
3.19.1.2 head . . . 27
3.19.1.3 tail . . . 27
3.20 page_dir Struct Reference . . . 28
3.20.1 Field Documentation . . . 28
3.20.1.1 tables . . . 28
3.20.1.2 tables_phys . . . 28
3.21 page_entry Struct Reference . . . 29
3.21.1 Field Documentation . . . 29
3.21.1.1 accessed . . . 29
3.21.1.2 dirty . . . 29
3.21.1.3 frameaddr . . . 29
3.21.1.4 present . . . 29
3.21.1.5 reserved . . . 29
3.21.1.6 usermode . . . 30
3.21.1.7 writeable . . . 30
3.22 page_table Struct Reference . . . 30
3.22.1 Field Documentation . . . 30
3.22.1.1 pages . . . 30
3.23 param Struct Reference . . . 31
3.23.1 Field Documentation . . . 31
3.23.1.1 buffer_ptr . . . 31
3.23.1.2 count_ptr . . . 31
3.23.1.3 device_id . . . 31
3.23.1.4 op_code . . . 31
3.24 PCB Struct Reference . . . 32
3.24.1 Field Documentation . . . 32
3.24.1.1 class . . . 32
3.24.1.2 name . . . 32
3.24.1.3 next . . . 32
3.24.1.4 prev . . . 33
3.24.1.5 priority . . . 33
3.24.1.6 stack . . . 33
3.24.1.7 stackBase . . . 33
3.24.1.8 stackTop . . . 33
3.24.1.9 status . . . 33
3.24.1.10 suspended . . . 33
3.25 Queue Struct Reference . . . 34
3.25.1 Field Documentation . . . 34
3.25.1.1 count . . . 34
3.25.1.2 head . . . 34

3.25.1.3 name . . . 34
3.25.1.4 tail . . . 35

4 File Documentation **37**

4.1 mpx_core/include/core/asm.h File Reference . . . 37
4.2 mpx_core/include/core/interrupts.h File Reference . . . 38
4.2.1 Function Documentation . . . 38
4.2.1.1 init_irq() . . . 38
4.2.1.2 init_pic() . . . 39
4.3 mpx_core/include/core/io.h File Reference . . . 40
4.3.1 Macro Definition Documentation . . . 40
4.3.1.1 inb . . . 40
4.3.1.2 outb . . . 40
4.4 mpx_core/include/core/serial.h File Reference . . . 40
4.4.1 Macro Definition Documentation . . . 41
4.4.1.1 COM1 . . . 41
4.4.1.2 COM2 . . . 42
4.4.1.3 COM3 . . . 42
4.4.1.4 COM4 . . . 42
4.4.2 Function Documentation . . . 42
4.4.2.1 addToHistory() . . . 42
4.4.2.2 backspaceChar() . . . 43
4.4.2.3 Function: backspaceChar . . . 43
4.4.2.4 comEmpty() . . . 43
4.4.2.5 deleteChar() . . . 43
4.4.2.6 Function: deleteChar . . . 43
4.4.2.7 getCommand() . . . 44
4.4.2.8 getCommandDown() . . . 44
4.4.2.9 getCommandUp() . . . 44
4.4.2.10 getHistory() . . . 44
4.4.2.11 historyFull() . . . 45
4.4.2.12 init_serial() . . . 45
4.4.2.13 initializeHistory() . . . 45
4.4.2.14 isBackspace() . . . 45
4.4.2.15 isDownArrow() . . . 45
4.4.2.16 isEnter() . . . 45
4.4.2.17 isEscape() . . . 45
4.4.2.18 isLeftArrow() . . . 46
4.4.2.19 isLetter() . . . 46
4.4.2.20 isRightArrow() . . . 46
4.4.2.21 isUpArrow() . . . 46
4.4.2.22 leftArrowChar() . . . 46

4.4.2.23 Function: leftArrowChar . . . 46
4.4.2.24 letterChar() . . . 47
4.4.2.25 Function: letterChar . . . 47
4.4.2.26 polling() . . . 47
4.4.2.27 Function: polling . . . 47
4.4.2.28 Function: polling . . . 48
4.4.2.29 resetCursor() . . . 49
4.4.2.30 Function: resetCursor . . . 50
4.4.2.31 Function: resetCursor . . . 50
4.4.2.32 resetLine() . . . 50
4.4.2.33 Function: resetLine . . . 50
4.4.2.34 Function: resetLine . . . 51
4.4.2.35 rightArrowChar() . . . 51
4.4.2.36 Function: righArrowChar . . . 51
4.4.2.37 serial_print() . . . 52
4.4.2.38 serial_println() . . . 52
4.4.2.39 set_serial_in() . . . 52
4.4.2.40 set_serial_out() . . . 52
4.5 mpx_core/include/core/tables.h File Reference . . . 52
4.5.1 Function Documentation . . . 53
4.5.1.1 __attribute__() . . . 53
4.5.1.2 gdt_init_entry() . . . 54
4.5.1.3 idt_set_gate() . . . 54
4.5.1.4 init_gdt() . . . 54
4.5.1.5 init_idt() . . . 55
4.5.2 Variable Documentation . . . 55
4.5.2.1 access . . . 55
4.5.2.2 base . . . 55
4.5.2.3 base_high . . . 55
4.5.2.4 base_low . . . 55
4.5.2.5 base_mid . . . 56
4.5.2.6 flags . . . 56
4.5.2.7 limit . . . 56
4.5.2.8 limit_low . . . 56
4.5.2.9 sselect . . . 56
4.5.2.10 zero . . . 56
4.6 mpx_core/include/mem/heap.h File Reference . . . 56
4.6.1 Macro Definition Documentation . . . 57
4.6.1.1 KHEAP_BASE . . . 57
4.6.1.2 KHEAP_MIN . . . 57
4.6.1.3 KHEAP_SIZE . . . 57
4.6.1.4 TABLE_SIZE . . . 58

4.6.2 Function Documentation . . . 58
4.6.2.1 _kmalloc() . . . 58
4.6.2.2 alloc() . . . 58
4.6.2.3 init_kheap() . . . 59
4.6.2.4 kfree() . . . 59
4.6.2.5 kmalloc() . . . 59
4.6.2.6 make_heap() . . . 59
4.7 mpx_core/include/mem/paging.h File Reference . . . 60
4.7.1 Macro Definition Documentation . . . 61
4.7.1.1 PAGE_SIZE . . . 61
4.7.2 Function Documentation . . . 61
4.7.2.1 clear_bit() . . . 61
4.7.2.2 first_free() . . . 61
4.7.2.3 get_bit() . . . 61
4.7.2.4 get_page() . . . 61
4.7.2.5 init_paging() . . . 62
4.7.2.6 load_page_dir() . . . 62
4.7.2.7 new_frame() . . . 62
4.7.2.8 set_bit() . . . 63
4.8 mpx_core/include/string.h File Reference . . . 63
4.8.1 Function Documentation . . . 63
4.8.1.1 atoi() . . . 64
4.8.1.2 isspace() . . . 64
4.8.1.3 itoa() . . . 64
4.8.1.4 Function: itoa . . . 64
4.8.1.5 memset() . . . 65
4.8.1.6 strcasecmp() . . . 65
4.8.1.7 Function: strcasecmp . . . 65
4.8.1.8 strcat() . . . 66
4.8.1.9 strcmp() . . . 66
4.8.1.10 strcpy() . . . 66
4.8.1.11 strlen() . . . 66
4.8.1.12 strtok() . . . 66
4.8.1.13 toupper() . . . 66
4.8.1.14 Function: toupper . . . 67
4.9 mpx_core/include/system.h File Reference . . . 67
4.9.1 Macro Definition Documentation . . . 68
4.9.1.1 asm . . . 68
4.9.1.2 cli . . . 68
4.9.1.3 GDT_CS_ID . . . 68
4.9.1.4 GDT_DS_ID . . . 68
4.9.1.5 hlt . . . 68

4.9.1.6 iret . . . 68
4.9.1.7 no_warn . . . 68
4.9.1.8 nop . . . 69
4.9.1.9 NULL . . . 69
4.9.1.10 sti . . . 69
4.9.1.11 volatile . . . 69
4.9.2 Typedef Documentation . . . 69
4.9.2.1 size_t . . . 69
4.9.2.2 u16int . . . 69
4.9.2.3 u32int . . . 69
4.9.2.4 u8int . . . 70
4.9.3 Function Documentation . . . 70
4.9.3.1 klogv() . . . 70
4.9.3.2 kpanic() . . . 70
4.10 mpx_core/kernel/core/interrupts.c File Reference . . . 71
4.10.1 Macro Definition Documentation . . . 72
4.10.1.1 ICW1 . . . 72
4.10.1.2 ICW4 . . . 72
4.10.1.3 io_wait . . . 72
4.10.1.4 PIC1 . . . 73
4.10.1.5 PIC2 . . . 73
4.10.2 Function Documentation . . . 73
4.10.2.1 bounds() . . . 73
4.10.2.2 breakpoint() . . . 73
4.10.2.3 coprocessor() . . . 73
4.10.2.4 coprocessor_segment() . . . 73
4.10.2.5 debug() . . . 73
4.10.2.6 device_not_available() . . . 74
4.10.2.7 divide_error() . . . 74
4.10.2.8 do_bounds() . . . 74
4.10.2.9 do_breakpoint() . . . 74
4.10.2.10 do_coprocessor() . . . 75
4.10.2.11 do_coprocessor_segment() . . . 75
4.10.2.12 do_debug() . . . 75
4.10.2.13 do_device_not_available() . . . 76
4.10.2.14 do_divide_error() . . . 76
4.10.2.15 do_double_fault() . . . 76
4.10.2.16 do_general_protection() . . . 77
4.10.2.17 do_invalid_op() . . . 77
4.10.2.18 do_invalid_tss() . . . 77
4.10.2.19 do_isr() . . . 78
4.10.2.20 do_nmi() . . . 78

4.10.2.21 do_overflow() . . . 78
4.10.2.22 do_page_fault() . . . 79
4.10.2.23 do_reserved() . . . 79
4.10.2.24 do_segment_not_present() . . . 79
4.10.2.25 do_stack_segment() . . . 80
4.10.2.26 double_fault() . . . 80
4.10.2.27 general_protection() . . . 80
4.10.2.28 init_irq() . . . 80
4.10.2.29 init_pic() . . . 81
4.10.2.30 invalid_op() . . . 82
4.10.2.31 invalid_tss() . . . 82
4.10.2.32 isr0() . . . 82
4.10.2.33 nmi() . . . 82
4.10.2.34 overflow() . . . 82
4.10.2.35 page_fault() . . . 82
4.10.2.36 reserved() . . . 82
4.10.2.37 rtc_isr() . . . 82
4.10.2.38 segment_not_present() . . . 83
4.10.2.39 serial_io_isr() . . . 83
4.10.2.40 stack_segment() . . . 83
4.10.2.41 sys_call_isr() . . . 83
4.10.3 Variable Documentation . . . 83
4.10.3.1 idt_entries . . . 83
4.11 mpx_core/kernel/core/kmain.c File Reference . . . 83
4.11.1 Function Documentation . . . 84
4.11.1.1 kmain() . . . 84
4.12 mpx_core/kernel/core/serial.c File Reference . . . 85
4.12.1 Macro Definition Documentation . . . 87
4.12.1.1 BACKSPACE . . . 87
4.12.1.2 CLEAR_LINE . . . 87
4.12.1.3 DELETE . . . 87
4.12.1.4 DOWN . . . 88
4.12.1.5 DOWN_ARROW . . . 88
4.12.1.6 ENTER . . . 88
4.12.1.7 ESCAPE . . . 88
4.12.1.8 LEFT_ARROW . . . 88
4.12.1.9 LEFT_BRACKET . . . 88
4.12.1.10 max . . . 88
4.12.1.11 MAX_SIZE . . . 88
4.12.1.12 MV_CURSOR_LEFT . . . 89
4.12.1.13 MV_CURSOR_RIGHT . . . 89
4.12.1.14 MV_CURSOR_START . . . 89

4.12.1.15 NEWLINE . . . 89
4.12.1.16 NO_ERROR . . . 89
4.12.1.17 PRINT_GREEN . . . 89
4.12.1.18 RIGHT_ARROW . . . 89
4.12.1.19 UP . . . 89
4.12.1.20 UP_ARROW . . . 90
4.12.2 Function Documentation . . . 90
4.12.2.1 addToHistory() . . . 90
4.12.2.2 backspaceChar() . . . 90
4.12.2.3 Function: backspaceChar . . . 90
4.12.2.4 comEmpty() . . . 91
4.12.2.5 deleteChar() . . . 91
4.12.2.6 getCommandDown() . . . 92
4.12.2.7 getCommandUp() . . . 92
4.12.2.8 getHistory() . . . 92
4.12.2.9 historyFull() . . . 93
4.12.2.10 init_serial() . . . 93
4.12.2.11 initializeHistory() . . . 93
4.12.2.12 isBackspace() . . . 93
4.12.2.13 isDownArrow() . . . 93
4.12.2.14 isEnter() . . . 93
4.12.2.15 isEscape() . . . 93
4.12.2.16 isLeftArrow() . . . 94
4.12.2.17 isLetter() . . . 94
4.12.2.18 isRightArrow() . . . 94
4.12.2.19 isUpArrow() . . . 94
4.12.2.20 leftArrowChar() . . . 94
4.12.2.21 Function: leftArrowChar . . . 94
4.12.2.22 letterChar() . . . 95
4.12.2.23 polling() . . . 95
4.12.2.24 Function: polling . . . 95
4.12.2.25 resetCursor() . . . 96
4.12.2.26 Function: resetCursor . . . 97
4.12.2.27 resetLine() . . . 97
4.12.2.28 Function: resetLine . . . 97
4.12.2.29 rightArrowChar() . . . 97
4.12.2.30 Function: righArrowChar . . . 97
4.12.2.31 serial_print() . . . 98
4.12.2.32 serial_println() . . . 98
4.12.2.33 set_serial_in() . . . 98
4.12.2.34 set_serial_out() . . . 98
4.12.3 Variable Documentation . . . 99

4.12.3.1 bufferIndex . . . 99
4.12.3.2 bufferSize . . . 99
4.12.3.3 commands . . . 99
4.12.3.4 index . . . 99
4.12.3.5 serial_port_in . . . 99
4.12.3.6 serial_port_out . . . 99
4.12.3.7 size . . . 99
4.13 mpx_core/kernel/core/system.c File Reference . . . 100
4.13.1 Function Documentation . . . 100
4.13.1.1 klogv() . . . 100
4.13.1.2 kpanic() . . . 101
4.14 mpx_core/kernel/core/tables.c File Reference . . . 101
4.14.1 Function Documentation . . . 102
4.14.1.1 gdt_init_entry() . . . 102
4.14.1.2 idt_set_gate() . . . 102
4.14.1.3 init_gdt() . . . 102
4.14.1.4 init_idt() . . . 103
4.14.1.5 write_gdt_ptr() . . . 103
4.14.1.6 write_idt_ptr() . . . 103
4.14.2 Variable Documentation . . . 103
4.14.2.1 gdt_entries . . . 103
4.14.2.2 gdt_ptr . . . 103
4.14.2.3 idt_entries . . . 104
4.14.2.4 idt_ptr . . . 104
4.15 mpx_core/kernel/mem/heap.c File Reference . . . 104
4.15.1 Function Documentation . . . 105
4.15.1.1 _kmalloc() . . . 105
4.15.1.2 alloc() . . . 105
4.15.1.3 kmalloc() . . . 106
4.15.1.4 make_heap() . . . 106
4.15.2 Variable Documentation . . . 106
4.15.2.1 __end . . . 106
4.15.2.2 _end . . . 106
4.15.2.3 curr_heap . . . 107
4.15.2.4 end . . . 107
4.15.2.5 kdir . . . 107
4.15.2.6 kheap . . . 107
4.15.2.7 phys_alloc_addr . . . 107
4.16 mpx_core/kernel/mem/paging.c File Reference . . . 107
4.16.1 Function Documentation . . . 108
4.16.1.1 clear_bit() . . . 108
4.16.1.2 find_free() . . . 108

4.16.1.3 get_bit() . . . 108
4.16.1.4 get_page() . . . 109
4.16.1.5 init_paging() . . . 109
4.16.1.6 load_page_dir() . . . 109
4.16.1.7 new_frame() . . . 110
4.16.1.8 set_bit() . . . 110
4.16.2 Variable Documentation . . . 110
4.16.2.1 cdir . . . 110
4.16.2.2 frames . . . 110
4.16.2.3 kdir . . . 110
4.16.2.4 kheap . . . 111
4.16.2.5 mem_size . . . 111
4.16.2.6 nframes . . . 111
4.16.2.7 page_size . . . 111
4.16.2.8 phys_alloc_addr . . . 111
4.17 mpx_core/lib/string.c File Reference . . . 111
4.17.1 Function Documentation . . . 112
4.17.1.1 abs() . . . 112
4.17.1.2 atoi() . . . 112
4.17.1.3 isspace() . . . 113
4.17.1.4 itoa() . . . 113
4.17.1.5 Function: itoa . . . 113
4.17.1.6 memset() . . . 114
4.17.1.7 reverse() . . . 114
4.17.1.8 strcasecmp() . . . 114
4.17.1.9 Function: strcasecmp . . . 114
4.17.1.10 strcat() . . . 115
4.17.1.11 strcmp() . . . 115
4.17.1.12 strcpy() . . . 115
4.17.1.13 strlen() . . . 115
4.17.1.14 strtok() . . . 115
4.17.1.15 swap() . . . 115
4.17.1.16 toupper() . . . 116
4.17.1.17 Function: toupper . . . 116
4.18 mpx_core/modules/mpx_supt.c File Reference . . . 116
4.18.1 Function Documentation . . . 117
4.18.1.1 idle() . . . 117
4.18.1.2 is_io_module_active() . . . 117
4.18.1.3 mpx_init() . . . 117
4.18.1.4 printlnMessage() . . . 118
4.18.1.5 Function: printMessage . . . 118
4.18.1.6 printMessage() . . . 118

4.18.1.7 Function: printMessage . . . 118
4.18.1.8 sys_alloc_mem() . . . 119
4.18.1.9 sys_free_mem() . . . 119
4.18.1.10 sys_req() . . . 119
4.18.1.11 sys_set_free() . . . 119
4.18.1.12 sys_set_malloc() . . . 120
4.18.2 Variable Documentation . . . 120
4.18.2.1 current_module . . . 120
4.18.2.2 params . . . 120
4.18.2.3 student_free . . . 120
4.18.2.4 student_malloc . . . 120
4.19 mpx_core/modules/mpx_supt.h File Reference . . . 121
4.19.1 Macro Definition Documentation . . . 122
4.19.1.1 COM_PORT . . . 122
4.19.1.2 DEFAULT_DEVICE . . . 122
4.19.1.3 EXIT . . . 122
4.19.1.4 FALSE . . . 122
4.19.1.5 IDLE . . . 122
4.19.1.6 INVALID_BUFFER . . . 123
4.19.1.7 INVALID_COUNT . . . 123
4.19.1.8 INVALID_OPERATION . . . 123
4.19.1.9 IO_MODULE . . . 123
4.19.1.10 MEM_MODULE . . . 123
4.19.1.11 MODULE_F . . . 123
4.19.1.12 MODULE_R1 . . . 123
4.19.1.13 MODULE_R2 . . . 123
4.19.1.14 MODULE_R3 . . . 124
4.19.1.15 MODULE_R4 . . . 124
4.19.1.16 MODULE_R5 . . . 124
4.19.1.17 NO_ERROR . . . 124
4.19.1.18 READ . . . 124
4.19.1.19 TRUE . . . 124
4.19.1.20 WRITE . . . 124
4.19.2 Function Documentation . . . 124
4.19.2.1 idle() . . . 125
4.19.2.2 is_io_module_active() . . . 125
4.19.2.3 mpx_init() . . . 125
4.19.2.4 printlnMessage() . . . 125
4.19.2.5 Function: printMessage . . . 125
4.19.2.6 printMessage() . . . 126
4.19.2.7 Function: printMessage . . . 126
4.19.2.8 sys_alloc_mem() . . . 127

4.19.2.9 sys_free_mem() . . . 127
4.19.2.10 sys_req() . . . 127
4.19.2.11 sys_set_free() . . . 127
4.19.2.12 sys_set_malloc() . . . 128
4.20 mpx_core/modules/R1/comhand.c File Reference . . . 128
4.20.1 Macro Definition Documentation . . . 129
4.20.1.1 DOWN . . . 129
4.20.1.2 GREEN . . . 129
4.20.1.3 RED . . . 129
4.20.1.4 UP . . . 129
4.20.1.5 WHITE . . . 129
4.20.2 Function Documentation . . . 129
4.20.2.1 comhand() . . . 130
4.20.2.2 Function: comhand . . . 130
4.20.2.3 noSuchCommand() . . . 131
4.20.2.4 poll() . . . 132
4.20.2.5 Function: poll . . . 132
4.20.2.6 printGreen() . . . 133
4.20.2.7 printRed() . . . 133
4.20.2.8 printWhite() . . . 133
4.20.2.9 resetBuffer() . . . 134
4.20.3 Variable Documentation . . . 134
4.20.3.1 cmdBuffer . . . 134
4.20.3.2 cmdBufferSize . . . 134
4.20.3.3 intro . . . 134
4.20.3.4 introSize . . . 134
4.21 mpx_core/modules/R1/comhand.h File Reference . . . 135
4.21.1 Function Documentation . . . 135
4.21.1.1 comhand() . . . 136
4.21.1.2 Function: comhand . . . 136
4.21.1.3 Function: comhand . . . 136
4.21.1.4 createPCBC() . . . 138
4.21.1.5 deletePCBCall() . . . 138
4.21.1.6 Function: noSuchCommand . . . 138
4.21.1.7 noSuchCommand() . . . 138
4.21.1.8 poll() . . . 138
4.21.1.9 Function: poll . . . 139
4.21.1.10 printGreen() . . . 139
4.21.1.11 printRed() . . . 139
4.21.1.12 printWhite() . . . 140
4.21.1.13 resetBuffer() . . . 140
4.22 mpx_core/modules/R1/date.c File Reference . . . 140

4.22.1 Function Documentation . . . 141
4.22.1.1 getDate() . . . 141
4.22.1.2 Function: getDate . . . 141
4.22.1.3 parseDate() . . . 142
4.22.1.4 Function: parseDate . . . 142
4.22.1.5 setDate() . . . 142
4.22.1.6 Function: setDate . . . 142
4.22.1.7 setDateCall() . . . 143
4.23 mpx_core/modules/R1/date.h File Reference . . . 143
4.23.1 Function Documentation . . . 144
4.23.1.1 getDate() . . . 144
4.23.1.2 Function: getDate . . . 144
4.23.1.3 parseDate() . . . 144
4.23.1.4 Function: parseDate . . . 144
4.23.1.5 setDate() . . . 145
4.23.1.6 Function: setDate . . . 145
4.23.1.7 setDateCall() . . . 145
4.24 mpx_core/modules/R1/miscCommands.c File Reference . . . 146
4.24.1 Macro Definition Documentation . . . 147
4.24.1.1 CLEAR_ALL . . . 147
4.24.1.2 MOVE_DEFAULT . . . 147
4.24.2 Function Documentation . . . 147
4.24.2.1 clear() . . . 147
4.24.2.2 getTimeCall() . . . 148
4.24.2.3 help() . . . 148
4.24.2.4 Function: help . . . 148
4.24.2.5 help2() . . . 149
4.24.2.6 help3() . . . 149
4.24.2.7 help4() . . . 149
4.24.2.8 help5() . . . 150
4.24.2.9 showPCBCall() . . . 150
4.24.2.10 shutdown() . . . 150
4.24.2.11 Function: shutdown . . . 151
4.24.2.12 version() . . . 151
4.24.2.13 Function: version . . . 151
4.25 mpx_core/modules/R1/miscCommands.h File Reference . . . 152
4.25.1 Function Documentation . . . 152
4.25.1.1 clear() . . . 152
4.25.1.2 getTimeCall() . . . 153
4.25.1.3 help() . . . 153
4.25.1.4 Function: help . . . 153
4.25.1.5 help2() . . . 154

4.25.1.6 help3() . . . 154
4.25.1.7 help4() . . . 154
4.25.1.8 help5() . . . 155
4.25.1.9 showPCBCall() . . . 155
4.25.1.10 shutdown() . . . 155
4.25.1.11 Function: shutdown . . . 156
4.25.1.12 version() . . . 156
4.25.1.13 Function: version . . . 156
4.26 mpx_core/modules/R1/time.c File Reference . . . 157
4.26.1 Function Documentation . . . 157
4.26.1.1 bcdToDec() . . . 157
4.26.1.2 getTime() . . . 157
4.26.1.3 Function: getTime . . . 158
4.26.1.4 parseTime() . . . 158
4.26.1.5 Function: parseTime . . . 158
4.26.1.6 setTime() . . . 159
4.26.1.7 Function: setTime . . . 159
4.26.1.8 setTimeCall() . . . 159
4.27 mpx_core/modules/R1/time.h File Reference . . . 160
4.27.1 Function Documentation . . . 161
4.27.1.1 bcdToDec() . . . 161
4.27.1.2 getTime() . . . 161
4.27.1.3 Function: getTime . . . 161
4.27.1.4 parseTime() . . . 162
4.27.1.5 Function: parseTime . . . 162
4.27.1.6 setTime() . . . 163
4.27.1.7 Function: setTime . . . 163
4.27.1.8 setTimeCall() . . . 164
4.28 mpx_core/modules/R2/pcb.c File Reference . . . 164
4.28.1 Function Documentation . . . 165
4.28.1.1 allocatePCB() . . . 165
4.28.1.2 Function: allocatePCB . . . 165
4.28.1.3 freePCB() . . . 165
4.28.1.4 Function: freePCB . . . 166
4.28.1.5 printPCB() . . . 166
4.28.1.6 Function: printPCB . . . 166
4.28.1.7 setupPCB() . . . 167
4.28.1.8 Function: setupPCB . . . 167
4.29 mpx_core/modules/R2/pcb.h File Reference . . . 168
4.29.1 Macro Definition Documentation . . . 169
4.29.1.1 APPLICATION . . . 169
4.29.1.2 BLOCKED . . . 169

4.29.1.3 NOTSUSPENDED . . . 169
4.29.1.4 READY . . . 169
4.29.1.5 RUNNING . . . 169
4.29.1.6 stackSize . . . 169
4.29.1.7 SUSPENDED . . . 169
4.29.1.8 SYSTEM . . . 170
4.29.2 Typedef Documentation . . . 170
4.29.2.1 PCB . . . 170
4.29.3 Function Documentation . . . 170
4.29.3.1 allocatePCB() . . . 170
4.29.3.2 Function: allocatePCB . . . 170
4.29.3.3 freePCB() . . . 171
4.29.3.4 Function: freePCB . . . 171
4.29.3.5 printPCB() . . . 171
4.29.3.6 Function: printPCB . . . 171
4.29.3.7 setupPCB() . . . 172
4.29.3.8 Function: setupPCB . . . 172
4.30 mpx_core/modules/R2/processManagement.c File Reference . . . 173
4.30.1 Function Documentation . . . 174
4.30.1.1 blockPCB() . . . 174
4.30.1.2 clearQueues() . . . 175
4.30.1.3 createPCB() . . . 175
4.30.1.4 Function: createPCB . . . 175
4.30.1.5 createQueue() . . . 175
4.30.1.6 Function: showBlocked . . . 176
4.30.1.7 deletePCB() . . . 176
4.30.1.8 getNextReady() . . . 177
4.30.1.9 initQueues() . . . 177
4.30.1.10 Function: initQueues . . . 178
4.30.1.11 insertPCB() . . . 178
4.30.1.12 noQueueError() . . . 179
4.30.1.13 removeBlocked() . . . 179
4.30.1.14 removeReady() . . . 179
4.30.1.15 resumePCB() . . . 180
4.30.1.16 setPCBpriority() . . . 180
4.30.1.17 showAll() . . . 181
4.30.1.18 showBlocked() . . . 181
4.30.1.19 showPCB() . . . 181
4.30.1.20 Function: showPCB . . . 181
4.30.1.21 showReady() . . . 182
4.30.1.22 Function: showReady . . . 182
4.30.1.23 suspendPCB() . . . 183

4.30.1.24 unblockPCB() . . . 183
4.30.2 Variable Documentation . . . 184
4.30.2.1 blockedQueue . . . 184
4.30.2.2 queuesCleared . . . 184
4.30.2.3 readyQueue . . . 184
4.31 mpx_core/modules/R2/processManagement.h File Reference . . . 184
4.31.1 Function Documentation . . . 185
4.31.1.1 blockPCB() . . . 186
4.31.1.2 clearQueues() . . . 186
4.31.1.3 createPCB() . . . 186
4.31.1.4 Function: createPCB . . . 186
4.31.1.5 createQueue() . . . 187
4.31.1.6 Function: showBlocked . . . 187
4.31.1.7 deletePCB() . . . 188
4.31.1.8 getNextReady() . . . 188
4.31.1.9 initQueues() . . . 189
4.31.1.10 Function: initQueues . . . 189
4.31.1.11 insertPCB() . . . 189
4.31.1.12 noQueueError() . . . 190
4.31.1.13 removeBlocked() . . . 190
4.31.1.14 removeReady() . . . 191
4.31.1.15 resumePCB() . . . 191
4.31.1.16 setPCBpriority() . . . 191
4.31.1.17 showAll() . . . 192
4.31.1.18 showBlocked() . . . 192
4.31.1.19 showPCB() . . . 193
4.31.1.20 Function: showPCB . . . 193
4.31.1.21 showReady() . . . 194
4.31.1.22 Function: showReady . . . 194
4.31.1.23 suspendPCB() . . . 194
4.31.1.24 unblockPCB() . . . 195
4.32 mpx_core/modules/R2/queue.c File Reference . . . 195
4.32.1 Function Documentation . . . 196
4.32.1.1 findPCB() . . . 196
4.32.1.2 Function: findPCB() . . . 196
4.32.1.3 isEmpty() . . . 197
4.32.1.4 peek() . . . 197
4.32.1.5 printQueue() . . . 197
4.32.1.6 Function: createQueue . . . 197
4.32.1.7 Function: findPCB() . . . 197
4.32.1.8 Function: printQueue . . . 198
4.32.1.9 queueInsert() . . . 198

4.32.1.10 Function: queueInsert . . . 198
4.32.1.11 queueRemove() . . . 199
4.32.2 Variable Documentation . . . 199
4.32.2.1 blockedTitle . . . 199
4.32.2.2 border . . . 199
4.32.2.3 borderSize . . . 199
4.32.2.4 readyTitle . . . 199
4.32.2.5 titleSize . . . 199
4.33 mpx_core/modules/R2/queue.h File Reference . . . 200
4.33.1 Typedef Documentation . . . 200
4.33.1.1 Queue . . . 200
4.33.2 Function Documentation . . . 200
4.33.2.1 findPCB() . . . 200
4.33.2.2 Function: findPCB() . . . 200
4.33.2.3 isEmpty() . . . 201
4.33.2.4 peek() . . . 201
4.33.2.5 printQueue() . . . 201
4.33.2.6 Function: createQueue . . . 201
4.33.2.7 Function: findPCB() . . . 202
4.33.2.8 Function: printQueue . . . 202
4.33.2.9 queueInsert() . . . 202
4.33.2.10 Function: queueInsert . . . 202
4.33.2.11 queueRemove() . . . 203
4.34 mpx_core/modules/R2/R2Commands.c File Reference . . . 203
4.34.1 Function Documentation . . . 204
4.34.1.1 blockPCBCall() . . . 204
4.34.1.2 createPCBCall() . . . 204
4.34.1.3 deletePCBCall() . . . 205
4.34.1.4 Function: noSuchCommand . . . 205
4.34.1.5 resumePCBCall() . . . 206
4.34.1.6 setPriorityCall() . . . 206
4.34.1.7 showReadyCall() . . . 206
4.34.1.8 suspendPCBCall() . . . 207
4.34.1.9 unblockPCBCall() . . . 207
4.35 mpx_core/modules/R2/R2Commands.h File Reference . . . 208
4.35.1 Function Documentation . . . 209
4.35.1.1 blockPCBCall() . . . 209
4.35.1.2 createPCBCall() . . . 209
4.35.1.3 deletePCBCall() . . . 210
4.35.1.4 Function: noSuchCommand . . . 210
4.35.1.5 resumePCBCall() . . . 210
4.35.1.6 setPriorityCall() . . . 211

4.35.1.7 showReadyCall() . . . 211
4.35.1.8 suspendPCBCall() . . . 211
4.35.1.9 unblockPCBCall() . . . 212
4.36 mpx_core/modules/R3/context.c File Reference . . . 212
4.36.1 Typedef Documentation . . . 213
4.36.1.1 func_ptr . . . 213
4.36.2 Function Documentation . . . 213
4.36.2.1 io_scheduler() . . . 213
4.36.2.2 loadProcess() . . . 214
4.36.2.3 Function: loadProcess . . . 214
4.36.2.4 loadr3() . . . 215
4.36.2.5 Function: loadr3 . . . 215
4.36.2.6 sys_call() . . . 216
4.36.2.7 Function: sys_call . . . 216
4.36.2.8 yield() . . . 216
4.36.2.9 Function: yield . . . 216
4.36.3 Variable Documentation . . . 216
4.36.3.1 cop . . . 216
4.36.3.2 DCB . . . 217
4.36.3.3 ioqueue . . . 217
4.36.3.4 old . . . 217
4.36.3.5 params . . . 217
4.37 mpx_core/modules/R3/context.h File Reference . . . 217
4.37.1 Typedef Documentation . . . 217
4.37.1.1 Context . . . 217
4.37.2 Function Documentation . . . 217
4.37.2.1 io_scheduler() . . . 218
4.37.2.2 loadr3() . . . 218
4.37.2.3 Function: loadr3 . . . 218
4.37.2.4 sys_call() . . . 219
4.37.2.5 Function: sys_call . . . 219
4.37.2.6 yield() . . . 220
4.37.2.7 Function: yield . . . 220
4.38 mpx_core/modules/R3/test.c File Reference . . . 220
4.38.1 Macro Definition Documentation . . . 221
4.38.1.1 RC_1 . . . 221
4.38.1.2 RC_2 . . . 221
4.38.1.3 RC_3 . . . 221
4.38.1.4 RC_4 . . . 221
4.38.1.5 RC_5 . . . 221
4.38.2 Function Documentation . . . 222
4.38.2.1 proc1() . . . 222

4.38.2.2 proc2() . . . 222
4.38.2.3 proc3() . . . 222
4.38.2.4 proc4() . . . 222
4.38.2.5 proc5() . . . 223
4.38.3 Variable Documentation . . . 223
4.38.3.1 er1 . . . 223
4.38.3.2 er2 . . . 223
4.38.3.3 er3 . . . 223
4.38.3.4 er4 . . . 223
4.38.3.5 er5 . . . 223
4.38.3.6 erSize . . . 224
4.38.3.7 msg1 . . . 224
4.38.3.8 msg2 . . . 224
4.38.3.9 msg3 . . . 224
4.38.3.10 msg4 . . . 224
4.38.3.11 msg5 . . . 224
4.38.3.12 msgSize . . . 224
4.39 mpx_core/modules/R3/test.h File Reference . . . 224
4.39.1 Function Documentation . . . 225
4.39.1.1 proc1() . . . 225
4.39.1.2 proc2() . . . 225
4.39.1.3 proc3() . . . 225
4.39.1.4 proc4() . . . 226
4.39.1.5 proc5() . . . 226
4.40 mpx_core/modules/R4/alarm.c File Reference . . . 226
4.40.1 Function Documentation . . . 227
4.40.1.1 createAlarm() . . . 227
4.40.1.2 Function: createAlarm . . . 227
4.40.1.3 createAlarmCall() . . . 228
4.40.1.4 Function: createAlarmCall . . . 228
4.40.1.5 createAList() . . . 229
4.40.1.6 Function: createAlarm . . . 229
4.40.1.7 infiniteProcess() . . . 229
4.40.1.8 timeDaemon() . . . 230
4.40.1.9 Function: timeDaemon . . . 230
4.40.2 Variable Documentation . . . 231
4.40.2.1 head_alarms . . . 231
4.41 mpx_core/modules/R4/alarm.h File Reference . . . 231
4.41.1 Typedef Documentation . . . 232
4.41.1.1 alarm_t . . . 232
4.41.2 Function Documentation . . . 232
4.41.2.1 createAlarm() . . . 232

4.41.2.2 Function: createAlarm . . . 232
4.41.2.3 createAlarmCall() . . . 232
4.41.2.4 Function: alarmCommand . . . 233
4.41.2.5 Function: createAlarmCall . . . 233
4.41.2.6 createAList() . . . 233
4.41.2.7 Function: createAlarm . . . 233
4.41.2.8 infiniteProcess() . . . 234
4.41.2.9 timeDaemon() . . . 234
4.41.2.10 Function: timeDaemon . . . 234
4.42 mpx_core/modules/R4/list.c File Reference . . . 235
4.42.1 Function Documentation . . . 236
4.42.1.1 aListInsert() . . . 236
4.42.1.2 aListRemove() . . . 236
4.43 mpx_core/modules/R4/list.h File Reference . . . 237
4.43.1 Typedef Documentation . . . 238
4.43.1.1 aList . . . 238
4.43.2 Function Documentation . . . 238
4.43.2.1 aListInsert() . . . 238
4.43.2.2 aListRemove() . . . 238
4.44 mpx_core/modules/R4/loadComhand.c File Reference . . . 238
4.44.1 Function Documentation . . . 239
4.44.1.1 loadComhand() . . . 239
4.44.1.2 Function: loadComhand . . . 239
4.44.1.3 loadIdle() . . . 240
4.44.1.4 Function: loadIdle . . . 241
4.44.1.5 loadInfiniteProcess() . . . 241
4.44.1.6 Function: loadInfiniteProcess . . . 241
4.44.1.7 loadTimeDaemon() . . . 242
4.44.1.8 Function: loadTimeDaemon . . . 242
4.45 mpx_core/modules/R4/loadComhand.h File Reference . . . 243
4.45.1 Function Documentation . . . 243
4.45.1.1 loadComhand() . . . 244
4.45.1.2 Function: loadComhand . . . 244
4.45.1.3 loadIdle() . . . 245
4.45.1.4 Function: loadIdle . . . 245
4.45.1.5 loadInfiniteProcess() . . . 245
4.45.1.6 Function: loadInfiniteProcess . . . 245
4.45.1.7 loadTimeDaemon() . . . 246
4.45.1.8 Function: loadTimeDaemon . . . 246
4.46 mpx_core/modules/R5/memoryCommands.c File Reference . . . 247
4.46.1 Function Documentation . . . 248
4.46.1.1 allocateMemoryCall() . . . 248

4.46.1.2 Function: allocateMemoryCall . . . 248
4.46.1.3 FreeMemoryCall() . . . 248
4.46.1.4 Function: freeMemoryCall . . . 248
4.46.1.5 isEmptyCall() . . . 249
4.46.1.6 Function: isEmptyCall . . . 249
4.47 mpx_core/modules/R5/memoryCommands.h File Reference . . . 250
4.47.1 Function Documentation . . . 250
4.47.1.1 allocateMemoryCall() . . . 250
4.47.1.2 Function: allocateMemoryCall . . . 250
4.47.1.3 FreeMemoryCall() . . . 251
4.47.1.4 Function: freeMemoryCall . . . 251
4.47.1.5 isEmptyCall() . . . 252
4.47.1.6 Function: isEmptyCall . . . 252
4.48 mpx_core/modules/R5/memoryManagment.c File Reference . . . 252
4.48.1 Function Documentation . . . 253
4.48.1.1 allocateMemory() . . . 253
4.48.1.2 Function: allocateMemory . . . 253
4.48.1.3 createmList() . . . 254
4.48.1.4 Function: createmList . . . 254
4.48.1.5 freeMem() . . . 254
4.48.1.6 Function: freeMem . . . 254
4.48.1.7 heapIsEmpty() . . . 255
4.48.1.8 Function: heapIsEmpty . . . 255
4.48.1.9 initializeHeap() . . . 255
4.48.1.10 Function: initializeHeap . . . 255
4.48.1.11 merge() . . . 255
4.48.1.12 Function: merge . . . 255
4.48.1.13 printAll() . . . 256
4.48.1.14 Function: printAll . . . 256
4.48.1.15 printAllocated() . . . 256
4.48.1.16 Function: printAllocated . . . 256
4.48.1.17 printCMCB() . . . 257
4.48.1.18 Function: printCMCB . . . 257
4.48.1.19 printFree() . . . 257
4.48.1.20 Function: printFree . . . 257
4.48.2 Variable Documentation . . . 258
4.48.2.1 heap_top . . . 258
4.48.2.2 MCBList . . . 258
4.49 mpx_core/modules/R5/memoryManagment.h File Reference . . . 258
4.49.1 Macro Definition Documentation . . . 259
4.49.1.1 ALLOCATED_MEMORY . . . 259
4.49.1.2 FREE_MEMORY . . . 259

4.49.1.3 HEAP_SIZE . . . 260
4.49.2 Typedef Documentation . . . 260
4.49.2.1 CMCB_t . . . 260
4.49.2.2 LMCB_t . . . 260
4.49.3 Function Documentation . . . 260
4.49.3.1 allocateMemory() . . . 260
4.49.3.2 Function: allocateMemory . . . 260
4.49.3.3 Function: allocateMemory . . . 260
4.49.3.4 createmList() . . . 261
4.49.3.5 Function: createmList . . . 261
4.49.3.6 freeMem() . . . 261
4.49.3.7 Function: freeMem . . . 261
4.49.3.8 Function: freeMem . . . 262
4.49.3.9 heapIsEmpty() . . . 262
4.49.3.10 Function: heapIsEmpty . . . 262
4.49.3.11 Function: heapIsEmpty . . . 262
4.49.3.12 initializeHeap() . . . 263
4.49.3.13 Function: initializeHeap . . . 263
4.49.3.14 Function: initializeHeap . . . 263
4.49.3.15 merge() . . . 263
4.49.3.16 Function: merge . . . 263
4.49.3.17 Function: merge . . . 263
4.49.3.18 printAll() . . . 264
4.49.3.19 Function: printAll . . . 264
4.49.3.20 printAllocated() . . . 264
4.49.3.21 Function: printAllocated . . . 264
4.49.3.22 printCMCB() . . . 265
4.49.3.23 Function: printCMCB . . . 265
4.49.3.24 printFree() . . . 265
4.49.3.25 Function: printFree . . . 265
4.50 mpx_core/modules/R5/mList.c File Reference . . . 266
4.50.1 Function Documentation . . . 266
4.50.1.1 mListInsert() . . . 267
4.50.1.2 Function: mListInsert . . . 267
4.50.1.3 mListRemove() . . . 267
4.50.1.4 Function: mListInsert . . . 267
4.51 mpx_core/modules/R5/mList.h File Reference . . . 267
4.51.1 Typedef Documentation . . . 268
4.51.1.1 mList . . . 268
4.51.2 Function Documentation . . . 269
4.51.2.1 mListInsert() . . . 269
4.51.2.2 Function: mListInsert . . . 269

4.51.2.3 mListRemove() . . . 269
4.51.2.4 Function: mListInsert . . . 269
4.52 mpx_core/modules/R6/driver.c File Reference . . . 269
4.52.1 Function Documentation . . . 270
4.52.1.1 com_close() . . . 270
4.52.1.2 com_open() . . . 271
4.52.1.3 com_read() . . . 271
4.52.1.4 com_write() . . . 272
4.52.1.5 disable_interrupts() . . . 272
4.52.1.6 enable_interrupts() . . . 273
4.52.1.7 pic_mask() . . . 273
4.52.1.8 serial_io() . . . 273
4.52.1.9 serial_read() . . . 273
4.52.1.10 serial_write() . . . 274
4.52.2 Variable Documentation . . . 274
4.52.2.1 DCB . . . 274
4.53 mpx_core/modules/R6/driver.h File Reference . . . 275
4.53.1 Macro Definition Documentation . . . 276
4.53.1.1 IRQ_COM1 . . . 276
4.53.1.2 PIC_EOI . . . 276
4.53.1.3 PIC_MASK . . . 276
4.53.1.4 PIC_REG . . . 276
4.53.1.5 REG_IER . . . 276
4.53.1.6 REG_IIR . . . 276
4.53.1.7 REG_LCR . . . 276
4.53.1.8 REG_LSB . . . 276
4.53.1.9 REG_LSR . . . 276
4.53.1.10 REG_MCR . . . 276
4.53.1.11 REG_MSB . . . 277
4.53.1.12 REG_MSR . . . 277
4.53.1.13 REG_RHR . . . 277
4.53.1.14 REG_SCRATCH . . . 277
4.53.1.15 REG_THR . . . 277
4.53.2 Typedef Documentation . . . 277
4.53.2.1 dcb_t . . . 277
4.53.2.2 status_t . . . 277
4.53.3 Enumeration Type Documentation . . . 277
4.53.3.1 status_t . . . 278
4.53.4 Function Documentation . . . 278
4.53.4.1 com_close() . . . 278
4.53.4.2 com_open() . . . 278
4.53.4.3 com_read() . . . 279

4.53.4.4 com_write() . . . 280
4.53.4.5 disable_interrupts() . . . 280
4.53.4.6 enable_interrupts() . . . 280
4.53.4.7 pic_mask() . . . 280
4.53.4.8 serial_io() . . . 280
4.53.4.9 serial_read() . . . 281
4.53.4.10 serial_write() . . . 281
4.54 mpx_core/modules/R6/IOqueue.c File Reference . . . 282
4.54.1 Function Documentation . . . 282
4.54.1.1 dequeue() . . . 283
4.54.1.2 enqueue() . . . 283
4.54.1.3 isEmpty_IO() . . . 283
4.54.1.4 peek_IO() . . . 283
4.54.2 Variable Documentation . . . 283
4.54.2.1 ioqueue . . . 283
4.55 mpx_core/modules/R6/IOqueue.h File Reference . . . 283
4.55.1 Typedef Documentation . . . 284
4.55.1.1 iod_t . . . 284
4.55.1.2 ioqueue_t . . . 284
4.55.2 Function Documentation . . . 284
4.55.2.1 dequeue() . . . 284
4.55.2.2 enqueue() . . . 285
4.55.2.3 isEmpty_IO() . . . 285
4.55.2.4 peek_IO() . . . 285

Index **287**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

alarm_s 5
aList 6
CMCB_s 7
Context 9
date_time 11
dcb_s 13
footer 15
gdt_descriptor_struct 16
gdt_entry_struct 16
header 17
heap 18
idt_entry_struct 19
idt_struct 21
index_entry 21
index_table 22
iod_s 23
ioqueue_s 25
LMCB_s 26
mList 27
page_dir 28
page_entry 29
page_table 30
param 31
PCB 32
Queue 34

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

mpx_core/include/string.h . . . 63
mpx_core/include/system.h . . . 67
mpx_core/include/core/asm.h . . . 37
mpx_core/include/core/interrupts.h . . . 38
mpx_core/include/core/io.h . . . 40
mpx_core/include/core/serial.h . . . 40
mpx_core/include/core/tables.h . . . 52
mpx_core/include/mem/heap.h . . . 56
mpx_core/include/mem/paging.h . . . 60
mpx_core/kernel/core/interrupts.c . . . 71
mpx_core/kernel/core/kmain.c . . . 83
mpx_core/kernel/core/serial.c . . . 85
mpx_core/kernel/core/system.c . . . 100
mpx_core/kernel/core/tables.c . . . 101
mpx_core/kernel/mem/heap.c . . . 104
mpx_core/kernel/mem/paging.c . . . 107
mpx_core/lib/string.c . . . 111
mpx_core/modules/mpx_supt.c . . . 116
mpx_core/modules/mpx_supt.h . . . 121
mpx_core/modules/R1/comhand.c . . . 128
mpx_core/modules/R1/comhand.h . . . 135
mpx_core/modules/R1/date.c . . . 140
mpx_core/modules/R1/date.h . . . 143
mpx_core/modules/R1/miscCommands.c . . . 146
mpx_core/modules/R1/miscCommands.h . . . 152
mpx_core/modules/R1/time.c . . . 157
mpx_core/modules/R1/time.h . . . 160
mpx_core/modules/R2/pcb.c . . . 164
mpx_core/modules/R2/pcb.h . . . 168
mpx_core/modules/R2/processManagement.c . . . 173
mpx_core/modules/R2/processManagement.h . . . 184
mpx_core/modules/R2/queue.c . . . 195
mpx_core/modules/R2/queue.h . . . 200
mpx_core/modules/R2/R2Commands.c . . . 203
mpx_core/modules/R2/R2Commands.h . . . 208

mpx_core/modules/R3/context.c . . . 212
mpx_core/modules/R3/context.h . . . 217
mpx_core/modules/R3/test.c . . . 220
mpx_core/modules/R3/test.h . . . 224
mpx_core/modules/R4/alarm.c . . . 226
mpx_core/modules/R4/alarm.h . . . 231
mpx_core/modules/R4/list.c . . . 235
mpx_core/modules/R4/list.h . . . 237
mpx_core/modules/R4/loadComhand.c . . . 238
mpx_core/modules/R4/loadComhand.h . . . 243
mpx_core/modules/R5/memoryCommands.c . . . 247
mpx_core/modules/R5/memoryCommands.h . . . 250
mpx_core/modules/R5/memoryManagment.c . . . 252
mpx_core/modules/R5/memoryManagment.h . . . 258
mpx_core/modules/R5/mList.c . . . 266
mpx_core/modules/R5/mList.h . . . 267
mpx_core/modules/R6/driver.c . . . 269
mpx_core/modules/R6/driver.h . . . 275
mpx_core/modules/R6/IOqueue.c . . . 282
mpx_core/modules/R6/IOqueue.h . . . 283

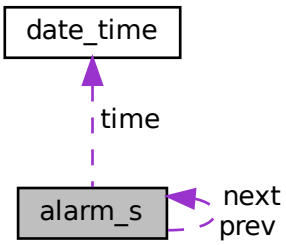
Chapter 3

Data Structure Documentation

3.1 alarm_s Struct Reference

`#include <alarm.h>`

Collaboration diagram for alarm_s:



Data Fields

- struct alarm_s * next
- struct alarm_s * prev
- date_time time
- char message [20]

3.1.1 Field Documentation

3.1.1.1 message

```
char message[20]
```

3.1.1.2 next

```
struct alarm_s* next
```

3.1.1.3 prev

```
struct alarm_s* prev
```

3.1.1.4 time

```
date_time time
```

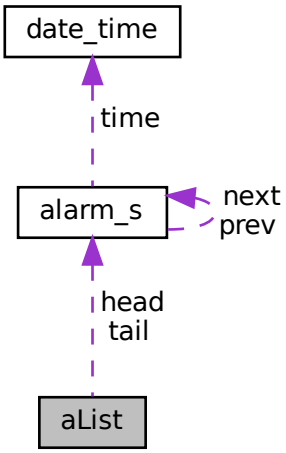
The documentation for this struct was generated from the following file:

- mpx_core/modules/R4/alarm.h

3.2 aList Struct Reference

```
#include <list.h>
```

Collaboration diagram for aList:



Data Fields

- int count
- alarm_t * head
- alarm_t * tail

3.2.1 Field Documentation

3.2.1.1 count

```
int count
```

3.2.1.2 head

```
alarm_t* head
```

3.2.1.3 tail

```
alarm_t* tail
```

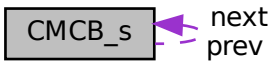
The documentation for this struct was generated from the following file:

- mpx_core/modules/R4/list.h

3.3 CMCB_s Struct Reference

```
#include <memoryManagment.h>
```

Collaboration diagram for CMCB_s:



Data Fields

- int type
- u32int start
- int size
- char namePCB [9]
- struct CMCB_s * next
- struct CMCB_s * prev

3.3.1 Field Documentation

3.3.1.1 namePCB

```
char namePCB[9]
```

3.3.1.2 next

```
struct CMCB_s* next
```

3.3.1.3 prev

```
struct CMCB_s* prev
```

3.3.1.4 size

```
int size
```

3.3.1.5 start

```
u32int start
```

3.3.1.6 type

```
int type
```

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

- mpx_core/modules/R5/memoryManagment.h

3.4 Context Struct Reference

```
#include <context.h>
```

Data Fields

- u32int gs
- u32int fs
- u32int es
- u32int ds
- u32int edi
- u32int esi
- u32int ebp
- u32int esp
- u32int ebx
- u32int edx
- u32int ecx
- u32int eax
- u32int eip
- u32int cs
- u32int eflags

3.4.1 Field Documentation

3.4.1.1 cs

```
u32int cs
```

3.4.1.2 ds

```
u32int ds
```

3.4.1.3 eax

```
u32int eax
```

3.4.1.4 ebp

```
u32int ebp
```

3.4.1.5 ebx

```
u32int ebx
```

3.4.1.6 ecx

```
u32int ecx
```

3.4.1.7 edi

```
u32int edi
```

3.4.1.8 edx

```
u32int edx
```

3.4.1.9 eflags

```
u32int eflags
```

3.4.1.10 eip

```
u32int eip
```

3.4.1.11 es

```
u32int es
```

3.4.1.12 esi

```
u32int esi
```

3.4.1.13 esp

```
u32int esp
```

3.4.1.14 fs

```
u32int fs
```

3.4.1.15 gs

```
u32int gs
```

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

- mpx_core/modules/R3/context.h

3.5 date_time Struct Reference

```
#include <system.h>
```

Data Fields

- int sec
- int min
- int hour
- int day_w
- int day_m
- int day_y
- int mon
- int year

3.5.1 Field Documentation

3.5.1.1 day_m

```
int day_m
```

3.5.1.2 day_w

```
int day_w
```

3.5.1.3 day_y

```
int day_y
```

3.5.1.4 hour

```
int hour
```

3.5.1.5 min

```
int min
```

3.5.1.6 mon

```
int mon
```

3.5.1.7 sec

```
int sec
```

3.5.1.8 year

```
int year
```

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

- mpx_core/include/system.h

3.6 dcb_s Struct Reference

```
#include <driver.h>
```

Data Fields

- int com_port
- int port_open
- int * e_flag
- status_t status
- char * buffer_ptr
- int * count_ptr
- char * buffer_loc
- int byte_count

3.6.1 Detailed Description

+* struct dcb represents a Device Control Block. +* A dcb should exist for each COM port, but you can just use COM1 +*

Parameters

com_port	the COM port. (You can omit this and just always use COM1) +*
port_open	whether the COM is open. +*
e_flag	whether the operation has completed (0 or 1). +*
status	the different operations (IDLE, READ, WRITE). +*
buffer_ptr	the buffer array to read into/write from. +*
count_ptr	how many characters to read/write. +*
buffer_loc	the current location we are reading/writing at. +*
byte_count	the number of bytes that have been read/written so far.

3.6.2 Field Documentation

3.6.2.1 buffer_loc

```
char* buffer_loc
```

3.6.2.2 buffer_ptr

```
char* buffer_ptr
```

3.6.2.3 byte_count

```
int byte_count
```

3.6.2.4 com_port

```
int com_port
```

3.6.2.5 count_ptr

```
int* count_ptr
```

3.6.2.6 e_flag

```
int* e_flag
```

3.6.2.7 port_open

```
int port_open
```

3.6.2.8 status

```
status_t status
```

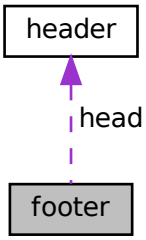
The documentation for this struct was generated from the following file:

- mpx_core/modules/R6/driver.h

3.7 footer Struct Reference

```
#include <heap.h>
```

Collaboration diagram for footer:



Data Fields

- header head

3.7.1 Field Documentation

3.7.1.1 head

```
header head
```

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

- mpx_core/include/mem/heap.h

3.8 gdt_descriptor_struct Struct Reference

`#include <tables.h>`

Data Fields

- u16int limit
- u32int base

3.8.1 Field Documentation

3.8.1.1 base

`u32int base`

3.8.1.2 limit

`u16int limit`

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

- mpx_core/include/core/tables.h

3.9 gdt_entry_struct Struct Reference

`#include <tables.h>`

Data Fields

- u16int limit_low
- u16int base_low
- u8int base_mid
- u8int access
- u8int flags
- u8int base_high

3.9.1 Field Documentation

3.9.1.1 access

```
u8int access
```

3.9.1.2 base_high

```
u8int base_high
```

3.9.1.3 base_low

```
u16int base_low
```

3.9.1.4 base_mid

```
u8int base_mid
```

3.9.1.5 flags

```
u8int flags
```

3.9.1.6 limit_low

```
u16int limit_low
```

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

- mpx_core/include/core/tables.h

3.10 header Struct Reference

```
#include <heap.h>
```

Data Fields

- int size
- int index_id

3.10.1 Field Documentation

3.10.1.1 index_id

```
int index_id
```

3.10.1.2 size

```
int size
```

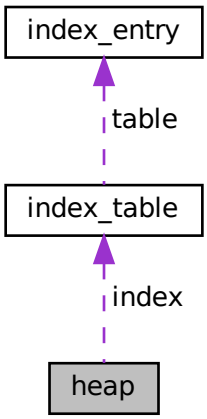
The documentation for this struct was generated from the following file:

- mpx_core/include/mem/heap.h

3.11 heap Struct Reference

```
#include <heap.h>
```

Collaboration diagram for heap:



Data Fields

- index_table index
- u32int base
- u32int max_size
- u32int min_size

3.11.1 Field Documentation

3.11.1.1 base

```
u32int base
```

3.11.1.2 index

```
index_table index
```

3.11.1.3 max_size

```
u32int max_size
```

3.11.1.4 min_size

```
u32int min_size
```

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

- mpx_core/include/mem/heap.h

3.12 idt_entry_struct Struct Reference

```
#include <tables.h>
```

Data Fields

- u16int base_low
- u16int sselect
- u8int zero
- u8int flags
- u16int base_high

3.12.1 Field Documentation

3.12.1.1 base_high

```
u16int base_high
```

3.12.1.2 base_low

```
u16int base_low
```

3.12.1.3 flags

```
u8int flags
```

3.12.1.4 sselect

```
u16int sselect
```

3.12.1.5 zero

```
u8int zero
```

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

- mpx_core/include/core/tables.h

3.13 idt_struct Struct Reference

`#include <tables.h>`

Data Fields

- u16int limit
- u32int base

3.13.1 Field Documentation

3.13.1.1 base

`u32int base`

3.13.1.2 limit

`u16int limit`

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

- mpx_core/include/core/tables.h

3.14 index_entry Struct Reference

`#include <heap.h>`

Data Fields

- int size
- int empty
- u32int block

3.14.1 Field Documentation

3.14.1.1 block

```
u32int block
```

3.14.1.2 empty

```
int empty
```

3.14.1.3 size

```
int size
```

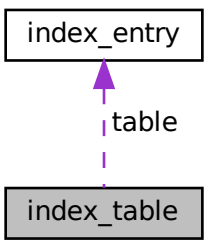
The documentation for this struct was generated from the following file:

- mpx_core/include/mem/heap.h

3.15 index_table Struct Reference

```
#include <heap.h>
```

Collaboration diagram for index_table:



Data Fields

- index_entry table [TABLE_SIZE]
- int id

3.15.1 Field Documentation

3.15.1.1 id

```
int id
```

3.15.1.2 table

```
index_entry table[TABLE_SIZE]
```

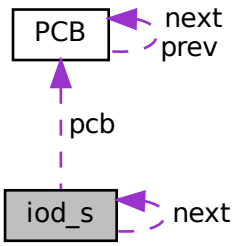
The documentation for this struct was generated from the following file:

- mpx_core/include/mem/heap.h

3.16 iod_s Struct Reference

```
#include <IOqueue.h>
```

Collaboration diagram for iod_s:



Data Fields

- int op_code
- char name [18]
- char * buffer_ptr
- int * count_ptr
- struct iod_s * next
- PCB * pcb

3.16.1 Field Documentation

3.16.1.1 buffer_ptr

```
char* buffer_ptr
```

3.16.1.2 count_ptr

```
int* count_ptr
```

3.16.1.3 name

```
char name[18]
```

3.16.1.4 next

```
struct iod_s* next
```

3.16.1.5 op_code

```
int op_code
```

3.16.1.6 pcb

```
PCB* pcb
```

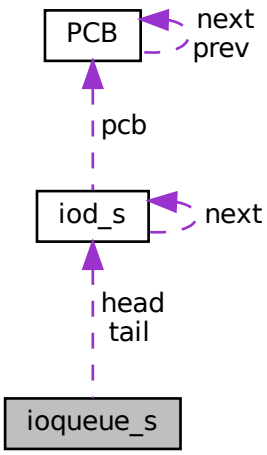
The documentation for this struct was generated from the following file:

- mpx_core/modules/R6/IOqueue.h

3.17 ioqueue_s Struct Reference

`#include <IOqueue.h>`

Collaboration diagram for ioqueue_s:



Data Fields

- iod_t * head
- iod_t * tail
- int count

3.17.1 Field Documentation

3.17.1.1 count

`int count`

3.17.1.2 head

`iod_t* head`

3.17.1.3 tail

```
iod_t* tail
```

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

- mpx_core/modules/R6/IOqueue.h

3.18 LMCB_s Struct Reference

```
#include <memoryManagment.h>
```

Data Fields

- int type
- int size

3.18.1 Field Documentation

3.18.1.1 size

```
int size
```

3.18.1.2 type

```
int type
```

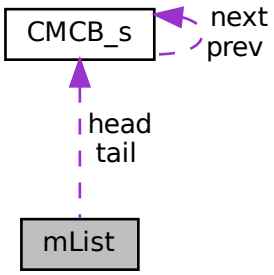
The documentation for this struct was generated from the following file:

- mpx_core/modules/R5/memoryManagment.h

3.19 mList Struct Reference

`#include <mList.h>`

Collaboration diagram for mList:



Data Fields

- int count
- CMCB_t * head
- CMCB_t * tail

3.19.1 Field Documentation

3.19.1.1 count

`int count`

3.19.1.2 head

`CMCB_t* head`

3.19.1.3 tail

`CMCB_t* tail`

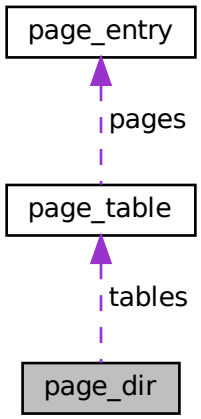
The documentation for this struct was generated from the following file:

- mpx_core/modules/R5/mList.h

3.20 page_dir Struct Reference

`#include <paging.h>`

Collaboration diagram for page_dir:



Data Fields

- page_table * tables [1024]
- u32int tables_phys [1024]

3.20.1 Field Documentation

3.20.1.1 tables

`page_table* tables[1024]`

3.20.1.2 tables_phys

`u32int tables_phys[1024]`

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

- mpx_core/include/mem/paging.h

3.21 page_entry Struct Reference

`#include <paging.h>`

Data Fields

- u32int present: 1
- u32int writeable: 1
- u32int usermode: 1
- u32int accessed: 1
- u32int dirty: 1
- u32int reserved: 7
- u32int frameaddr: 20

3.21.1 Field Documentation

3.21.1.1 accessed

`u32int accessed`

3.21.1.2 dirty

`u32int dirty`

3.21.1.3 frameaddr

`u32int frameaddr`

3.21.1.4 present

`u32int present`

3.21.1.5 reserved

`u32int reserved`

3.21.1.6 usermode

```
u32int usermode
```

3.21.1.7 writeable

```
u32int writeable
```

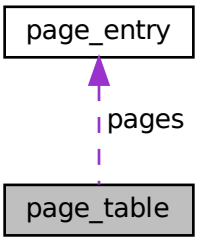
The documentation for this struct was generated from the following file:

- mpx_core/include/mem/paging.h

3.22 page_table Struct Reference

```
#include <paging.h>
```

Collaboration diagram for page_table:



Data Fields

- page_entry pages [1024]

3.22.1 Field Documentation

3.22.1.1 pages

```
page_entry pages[1024]
```

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

- mpx_core/include/mem/paging.h

3.23 param Struct Reference

```
#include <mpx_supt.h>
```

Data Fields

- int op_code
- int device_id
- char * buffer_ptr
- int * count_ptr

3.23.1 Field Documentation

3.23.1.1 buffer_ptr

```
char* buffer_ptr
```

3.23.1.2 count_ptr

```
int* count_ptr
```

3.23.1.3 device_id

```
int device_id
```

3.23.1.4 op_code

```
int op_code
```

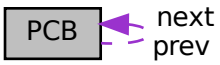
The documentation for this struct was generated from the following file:

- mpx_core/modules/mpx_supt.h

3.24 PCB Struct Reference

`#include <pcb.h>`

Collaboration diagram for PCB:



Data Fields

- char name [9]
- int class
- int priority
- int status
- int suspended
- struct PCB * next
- struct PCB * prev
- unsigned char stack [1024]
- unsigned char * stackTop
- unsigned char * stackBase

3.24.1 Field Documentation

3.24.1.1 class

`int class`

3.24.1.2 name

`char name[9]`

3.24.1.3 next

`struct PCB* next`

3.24.1.4 prev

```
struct PCB* prev
```

3.24.1.5 priority

```
int priority
```

3.24.1.6 stack

```
unsigned char stack[1024]
```

3.24.1.7 stackBase

```
unsigned char* stackBase
```

3.24.1.8 stackTop

```
unsigned char* stackTop
```

3.24.1.9 status

```
int status
```

3.24.1.10 suspended

```
int suspended
```

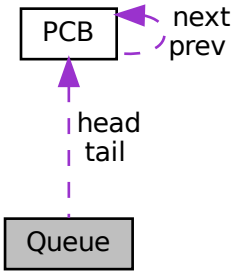
The documentation for this struct was generated from the following file:

- mpx_core/modules/R2/pcb.h

3.25 Queue Struct Reference

`#include <queue.h>`

Collaboration diagram for Queue:



Data Fields

- int count
- struct PCB * head
- struct PCB * tail
- char * name

3.25.1 Field Documentation

3.25.1.1 count

`int count`

3.25.1.2 head

`struct PCB* head`

3.25.1.3 name

`char* name`

3.25.1.4 tail

```
struct PCB* tail
```

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

- mpx_core/modules/R2/queue.h

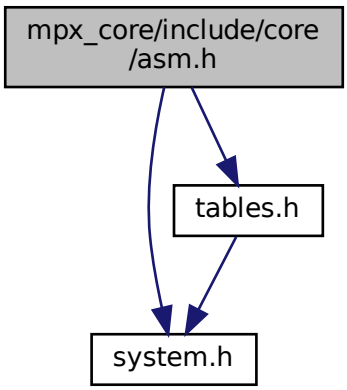
Chapter 4

File Documentation

4.1 mpx_core/include/core/asm.h File Reference

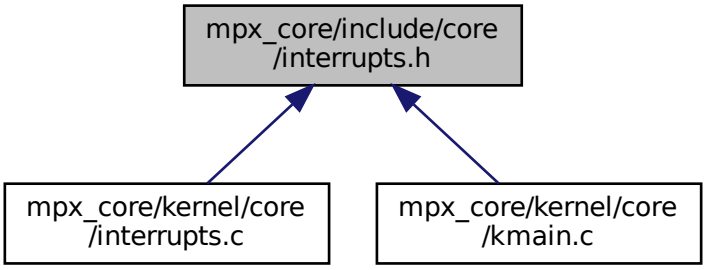
```
#include <system.h>
#include <tables.h>
```

Include dependency graph for asm.h:



4.2 mpx_core/include/core/interrupts.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

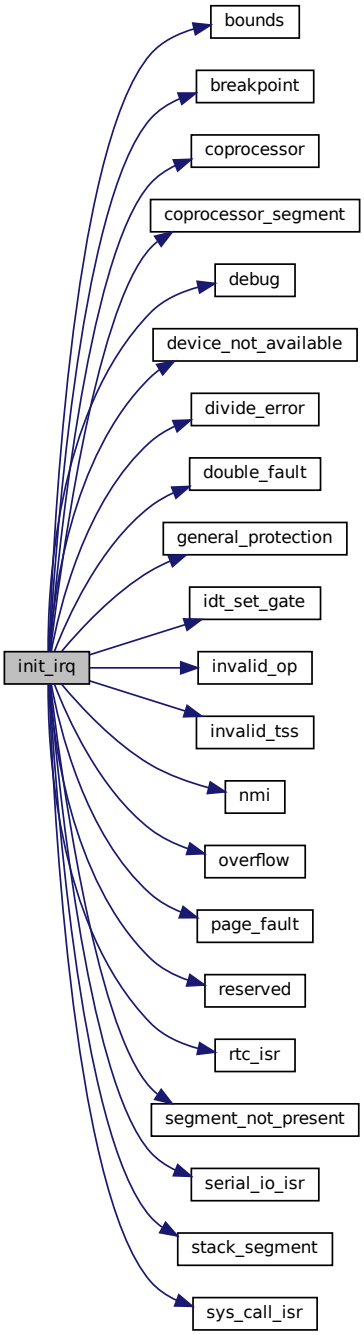
- void init_irq (void)
- void init_pic (void)

4.2.1 Function Documentation

4.2.1.1 init_irq()

```
void init_irq (
            void  )
```

Here is the call graph for this function:

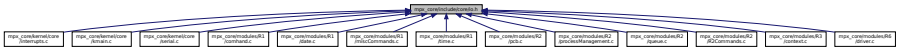


4.2.1.2 init_pic()

```
void init_pic (
            void  )
```

4.3 mpx_core/include/core/io.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define outb(port, data) asm volatile ("outb %%al,%%dx" : : "a" (data), "d" (port))
- #define inb(port)

4.3.1 Macro Definition Documentation

4.3.1.1 inb

```
#define inb(
               port )
```

Value:

```
      ({                          \
      unsigned char r;                    \
      asm volatile ("inb %%dx,%%al": "=a" (r): "d" (port)); \
      r;                                  \
    })
```

4.3.1.2 outb

```
#define outb(
               port,
               data )  asm volatile ("outb %%al,%%dx" :  :  "a" (data), "d" (port))
```

4.4 mpx_core/include/core/serial.h File Reference

This graph shows which files directly or indirectly include this file:

Macros

- #define COM1 0x3f8
- #define COM2 0x2f8
- #define COM3 0x3e8
- #define COM4 0x2e8

Functions

- int init_serial (int device)
- int serial_println (const char *msg)
- int serial_print (const char *msg)
- int set_serial_out (int device)
- int set_serial_in (int device)
- int * polling (char *buffer, int *count)
- void resetLine ()
- void resetCursor ()
- void backspaceChar ()
- void letterChar ()
- void deleteChar ()
- void rightArrowChar ()
- void leftArrowChar ()
- int isEscape (char *letter)
- int isBackspace (char *letter)
- int isEnter (char *letter)
- int isLetter (char *letter)
- int isRightArrow (char *letter)
- int isLeftArrow (char *letter)
- int isDownArrow (char *letter)
- int isUpArrow (char *letter)
- void initializeHistory ()
- int addToHistory (char *command)
- int historyFull ()
- char * getCommand ()
- int comEmpty ()
- void getHistory (char *buffer, char *letter)
- char * getCommandUp ()
- char * getCommandDown ()

4.4.1 Macro Definition Documentation

4.4.1.1 COM1

```
#define COM1 0x3f8
```

4.4.1.2 COM2

```
#define COM2 0x2f8
```

4.4.1.3 COM3

```
#define COM3 0x3e8
```

4.4.1.4 COM4

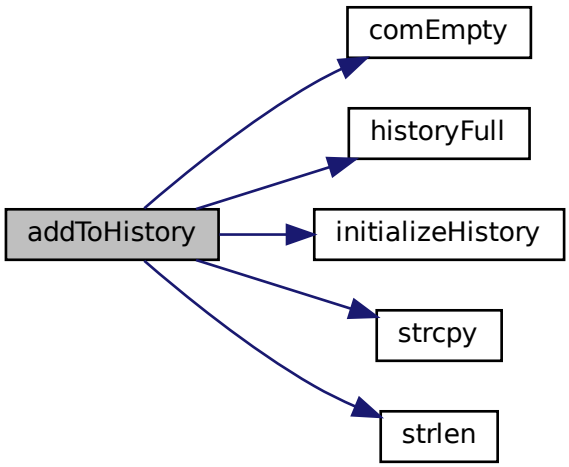
```
#define COM4 0x2e8
```

4.4.2 Function Documentation

4.4.2.1 addToHistory()

```
int addToHistory (
            char * command )
```

Here is the call graph for this function:



4.4.2.2 backspaceChar()

```
void backspaceChar ( )
```

4.4.2.3 Function: backspaceChar

Handles backspace characters. Copies the current buffers content up to the cursor location into a new buffer and concatenates that with everything past the cursor.

Parameters

∗buffer	the buffer to be changed

Author

Brendan Michael

4.4.2.4 comEmpty()

```
int comEmpty ( )
```

4.4.2.5 deleteChar()

```
void deleteChar ( )
```

4.4.2.6 Function: deleteChar

Handles delete character. Copies the current buffers content up to the cursor location into a new buffer and concatenates that with everything past the cursor not including the final character.

Parameters

∗buffer	the buffer to be changed

Author

Brendan Michael

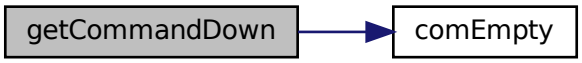
4.4.2.7 getCommand()

```
char* getCommand ( )
```

4.4.2.8 getCommandDown()

```
char* getCommandDown ( )
```

Here is the call graph for this function:



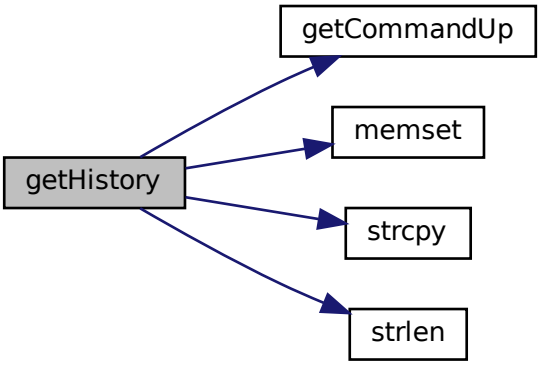
4.4.2.9 getCommandUp()

```
char* getCommandUp ( )
```

4.4.2.10 getHistory()

```
void getHistory (
            char * buffer,
            char * letter )
```

Here is the call graph for this function:



4.4.2.11 historyFull()

```
int historyFull ( )
```

4.4.2.12 init_serial()

```
int init_serial (
            int device )
```

4.4.2.13 initializeHistory()

```
void initializeHistory ( )
```

4.4.2.14 isBackspace()

```
int isBackspace (
            char * letter )
```

4.4.2.15 isDownArrow()

```
int isDownArrow (
            char * letter )
```

4.4.2.16 isEnter()

```
int isEnter (
            char * letter )
```

4.4.2.17 isEscape()

```
int isEscape (
            char * letter )
```

4.4.2.18 isLeftArrow()

```
int isLeftArrow (
            char * letter )
```

4.4.2.19 isLetter()

```
int isLetter (
            char * letter )
```

4.4.2.20 isRightArrow()

```
int isRightArrow (
            char * letter )
```

4.4.2.21 isUpArrow()

```
int isUpArrow (
            char * letter )
```

4.4.2.22 leftArrowChar()

```
void leftArrowChar ( )
```

4.4.2.23 Function: leftArrowChar

Moves the cursor one space to the left and decrements the buffer index.

Parameters

∗buffer	the buffer to be changed

Author

Brendan Michael

Here is the call graph for this function:

4.4.2.24 letterChar()

```
void letterChar ( )
```

4.4.2.25 Function: letterChar

Adds a non-special character to the buffer and adjusts buffer size and index.

Author

Brendan Michael

4.4.2.26 polling()

```
int* polling (
            char * buffer,
            int * count )
```

4.4.2.27 Function: polling

Fills the buffer with keyboard input and returns to command handler when enter key is pressed. Also handles special characters

Parameters

∗buffer	String buffer to be filled
∗count	size of the buffer

Returns

the buffer size

Author

Brendan Michael

4.4.2.28 Function: polling

Fills the buffer with keyboard input and returns to command handler when enter key is pressed. Also handles special characters

Parameters

∗*buffer*	String buffer to be filled
∗*count*	size of the buffer

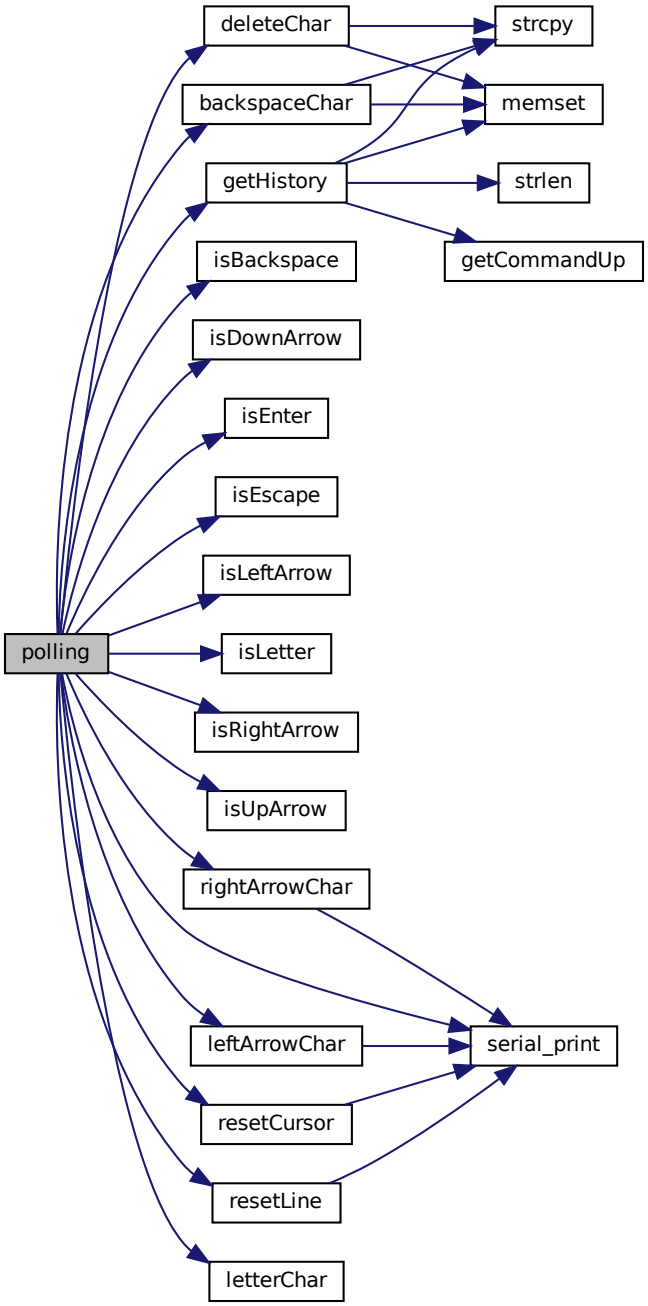
Returns

the buffer size

Author

Brendan Michael

Here is the call graph for this function:



4.4.2.29 resetCursor()

```
void resetCursor ( )
```

4.4.2.30 Function: resetCursor

Using escape sequences, moves the cursor from the start of the line up to the current location in the buffer

Author

Brendan Michael

4.4.2.31 Function: resetCursor

Using escape sequences, moves the cursor from the start of the line up to the current location in the buffer

Author

Brendan Michael

Here is the call graph for this function:

4.4.2.32 resetLine()

```
void resetLine ( )
```

4.4.2.33 Function: resetLine

Using escape sequences, moves the cursor to the beginning of the line, clears it, and changes the color to green.

Author

Brendan Michael

4.4.2.34 Function: resetLine

Using escape sequences, moves the cursor to the beginning of the line, clears it, and changes the color to green.

Author

Brendan Michael

Here is the call graph for this function:

4.4.2.35 rightArrowChar()

```
void rightArrowChar ( )
```

4.4.2.36 Function: righArrowChar

Moves the cursor one space to the right and increments the buffer index.

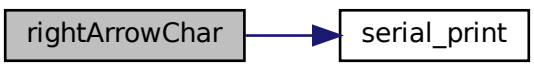
Parameters

∗buffer	the buffer to be changed

Author

Brendan Michael

Here is the call graph for this function:



4.4.2.37 serial_print()

```
int serial_print (
            const char * msg )
```

4.4.2.38 serial_println()

```
int serial_println (
            const char * msg )
```

4.4.2.39 set_serial_in()

```
int set_serial_in (
            int device )
```

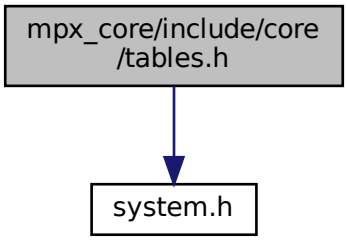
4.4.2.40 set_serial_out()

```
int set_serial_out (
            int device )
```

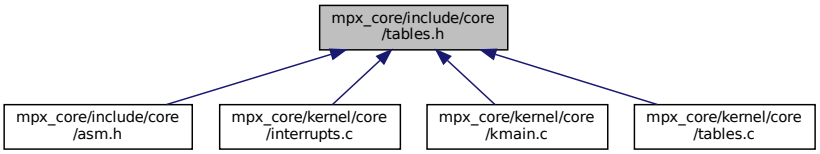
4.5 mpx_core/include/core/tables.h File Reference

```
#include "system.h"
```

Include dependency graph for tables.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct idt_entry_struct
- struct idt_struct
- struct gdt_descriptor_struct
- struct gdt_entry_struct

Functions

- struct idt_entry_struct __attribute__ ((packed)) idt_entry
- void idt_set_gate (u8int idx, u32int base, u16int sel, u8int flags)
- void gdt_init_entry (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void init_idt ()
- void init_gdt ()

Variables

- u16int base_low
- u16int sselect
- u8int zero
- u8int flags
- u16int base_high
- u16int limit
- u32int base
- u16int limit_low
- u8int base_mid
- u8int access

4.5.1 Function Documentation

4.5.1.1 __attribute__()

```
struct gdt_entry_struct __attribute__ (
             (packed)  )
```

4.5.1.2 gdt_init_entry()

```
void gdt_init_entry (
             int idx,
             u32int base,
             u32int limit,
             u8int access,
             u8int flags )
```

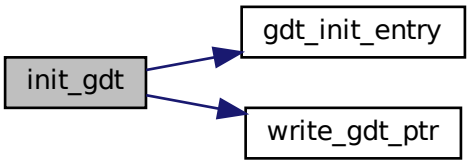
4.5.1.3 idt_set_gate()

```
void idt_set_gate (
             u8int idx,
             u32int base,
             u16int sel,
             u8int flags )
```

4.5.1.4 init_gdt()

```
void init_gdt ( )
```

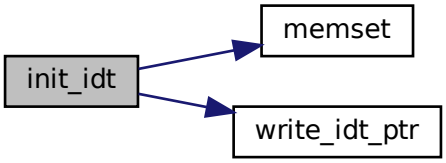
Here is the call graph for this function:



4.5.1.5 init_idt()

```
void init_idt ( )
```

Here is the call graph for this function:



4.5.2 Variable Documentation

4.5.2.1 access

```
u8int access
```

4.5.2.2 base

```
u32int base
```

4.5.2.3 base_high

```
u8int base_high
```

4.5.2.4 base_low

```
u16int base_low
```

4.5.2.5 base_mid

```
u8int base_mid
```

4.5.2.6 flags

```
u8int flags
```

4.5.2.7 limit

```
u16int limit
```

4.5.2.8 limit_low

```
u16int limit_low
```

4.5.2.9 sselect

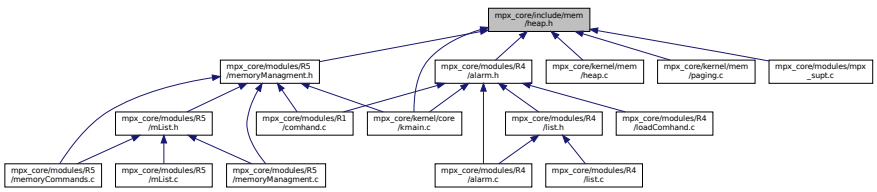
```
u16int sselect
```

4.5.2.10 zero

```
u8int zero
```

4.6 mpx_core/include/mem/heap.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct header
- struct footer
- struct index_entry
- struct index_table
- struct heap

Macros

- #define TABLE_SIZE 0x1000
- #define KHEAP_BASE 0xD000000
- #define KHEAP_MIN 0x10000
- #define KHEAP_SIZE 0x1000000

Functions

- u32int _kmalloc (u32int size, int align, u32int *phys_addr)
- u32int kmalloc (u32int size)
- u32int kfree ()
- void init_kheap ()
- u32int alloc (u32int size, heap *hp, int align)
- heap * make_heap (u32int base, u32int max, u32int min)

4.6.1 Macro Definition Documentation

4.6.1.1 KHEAP_BASE

```
#define KHEAP_BASE 0xD000000
```

4.6.1.2 KHEAP_MIN

```
#define KHEAP_MIN 0x10000
```

4.6.1.3 KHEAP_SIZE

```
#define KHEAP_SIZE 0x1000000
```

4.6.1.4 TABLE_SIZE

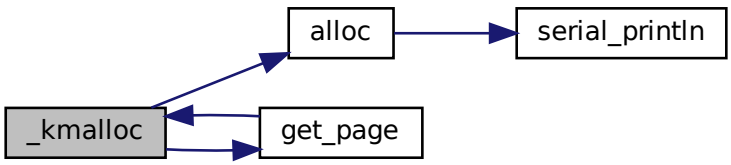
```
#define TABLE_SIZE 0x1000
```

4.6.2 Function Documentation

4.6.2.1 _kmalloc()

```
u32int _kmalloc (
            u32int size,
            int align,
            u32int * phys_addr )
```

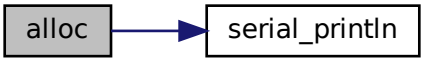
Here is the call graph for this function:



4.6.2.2 alloc()

```
u32int alloc (
            u32int size,
            heap * hp,
            int align )
```

Here is the call graph for this function:



4.6.2.3 init_kheap()

```
void init_kheap ( )
```

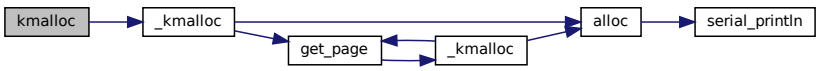
4.6.2.4 kfree()

```
u32int kfree ( )
```

4.6.2.5 kmalloc()

```
u32int kmalloc (
            u32int size )
```

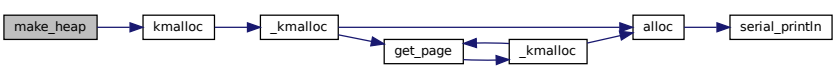
Here is the call graph for this function:



4.6.2.6 make_heap()

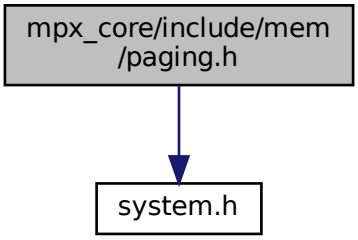
```
heap* make_heap (
            u32int base,
            u32int max,
            u32int min )
```

Here is the call graph for this function:

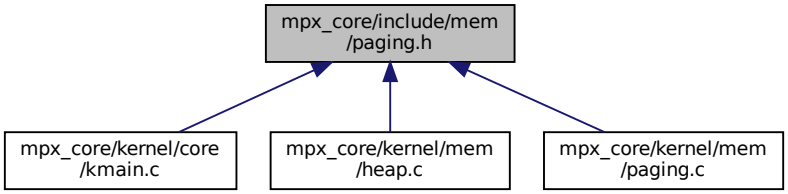


4.7 mpx_core/include/mem/paging.h File Reference

`#include <system.h>`
Include dependency graph for paging.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct page_entry
- struct page_table
- struct page_dir

Macros

- #define PAGE_SIZE 0x1000

Functions

- void set_bit (u32int addr)
- void clear_bit (u32int addr)
- u32int get_bit (u32int addr)
- u32int first_free ()
- void init_paging ()
- void load_page_dir (page_dir *new_page_dir)
- page_entry * get_page (u32int addr, page_dir *dir, int make_table)
- void new_frame (page_entry *page)

4.7.1 Macro Definition Documentation

4.7.1.1 PAGE_SIZE

```
#define PAGE_SIZE 0x1000
```

4.7.2 Function Documentation

4.7.2.1 clear_bit()

```
void clear_bit (
            u32int addr )
```

4.7.2.2 first_free()

```
u32int first_free ( )
```

4.7.2.3 get_bit()

```
u32int get_bit (
            u32int addr )
```

4.7.2.4 get_page()

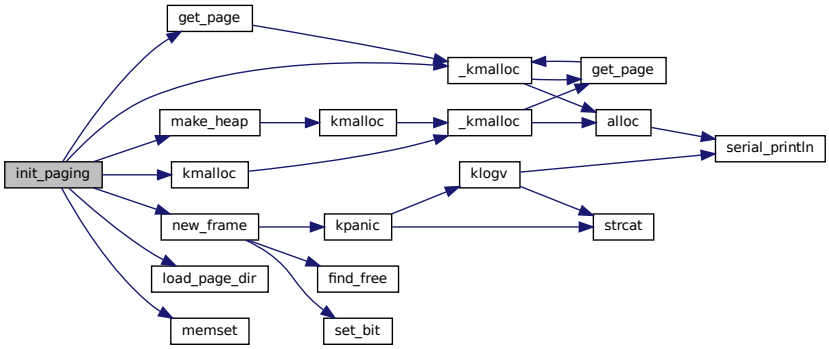
```
page_entry* get_page (
            u32int addr,
            page_dir * dir,
            int make_table )
```

Here is the call graph for this function:

4.7.2.5 init_paging()

```
void init_paging ( )
```

Here is the call graph for this function:



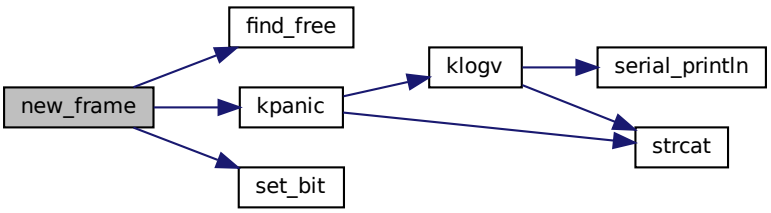
4.7.2.6 load_page_dir()

```
void load_page_dir (
            page_dir * new_page_dir )
```

4.7.2.7 new_frame()

```
void new_frame (
            page_entry * page )
```

Here is the call graph for this function:

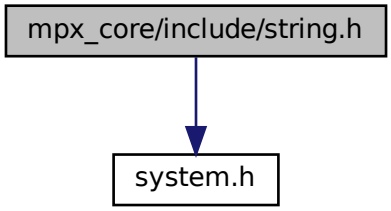


4.7.2.8 set_bit()

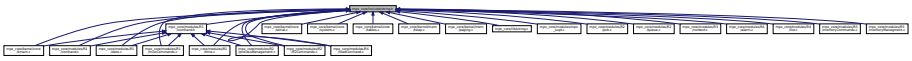
```
void set_bit (
              u32int addr )
```

4.8 mpx_core/include/string.h File Reference

`#include <system.h>`
Include dependency graph for string.h:



This graph shows which files directly or indirectly include this file:



Functions

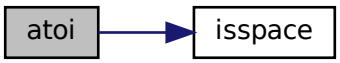
- int isspace (const char *c)
- void * memset (void *s, int c, size_t n)
- char * strcpy (char *s1, const char *s2)
- char * strcat (char *s1, const char *s2)
- int strlen (const char *s)
- int strcmp (const char *s1, const char *s2)
- char * strtok (char *s1, const char *s2)
- int atoi (const char *s)
- int strcasecmp (const char *s1, const char *s2)
- char toupper (char x)
- char * itoa (int value, char *buffer, int base)

4.8.1 Function Documentation

4.8.1.1 atoi()

```
int atoi (
            const char * s )
```

Here is the call graph for this function:



4.8.1.2 isspace()

```
int isspace (
            const char * c )
```

4.8.1.3 itoa()

```
char* itoa (
            int value,
            char * buffer,
            int base )
```

4.8.1.4 Function: itoa

Iterative function to implement itoa() function in C geeksforgeeks.org/implement-itoa/ for reference. Changed a little to make own.

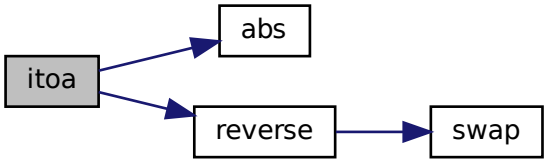
Parameters

value	integer taken as input
base	integer of the desired base to convert.
buffer	character buffer where the written output is being passed to.

Author

Bryce Williams

Here is the call graph for this function:



4.8.1.5 memset()

```
void* memset (
              void * s,
              int c,
              size_t n )
```

4.8.1.6 strcasecmp()

```
int strcasecmp (
              const char * s1,
              const char * s2 )
```

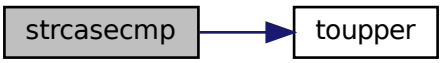
4.8.1.7 Function: strcasecmp

Compares two strings while being case insentsitive.

Author

Bryce Williams

Here is the call graph for this function:



4.8.1.8 strcat()

```
char* strcat (
            char * s1,
            const char * s2 )
```

4.8.1.9 strcmp()

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

4.8.1.10 strcpy()

```
char* strcpy (
            char * s1,
            const char * s2 )
```

4.8.1.11 strlen()

```
int strlen (
            const char * s )
```

4.8.1.12 strtok()

```
char* strtok (
            char * s1,
            const char * s2 )
```

4.8.1.13 toupper()

```
char toupper (
            char x )
```

4.8.1.14 Function: toupper

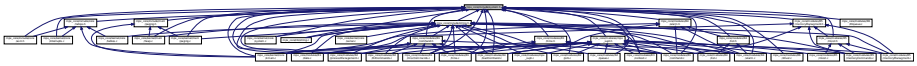
Converts character to uppercase. ex: 'a' -> 'A'

Author

Bryce Williams

4.9 mpx_core/include/system.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct date_time

Macros

- #define NULL 0
- #define no_warn(p) if (p) while (1) break
- #define asm __asm__
- #define volatile __volatile__
- #define sti() asm volatile ("sti"::)
- #define cli() asm volatile ("cli"::)
- #define nop() asm volatile ("nop"::)
- #define hlt() asm volatile ("hlt"::)
- #define iret() asm volatile ("iret"::)
- #define GDT_CS_ID 0x01
- #define GDT_DS_ID 0x02

Typedefs

- typedef unsigned int size_t
- typedef unsigned char u8int
- typedef unsigned short u16int
- typedef unsigned long u32int

Functions

- void klogv (const char *msg)
- void kpanic (const char *msg)

4.9.1 Macro Definition Documentation

4.9.1.1 asm

```
#define asm __asm__
```

4.9.1.2 cli

```
#define cli( ) asm volatile ("cli"::)
```

4.9.1.3 GDT_CS_ID

```
#define GDT_CS_ID 0x01
```

4.9.1.4 GDT_DS_ID

```
#define GDT_DS_ID 0x02
```

4.9.1.5 hlt

```
#define hlt( ) asm volatile ("hlt"::)
```

4.9.1.6 iret

```
#define iret( ) asm volatile ("iret"::)
```

4.9.1.7 no_warn

```
#define no_warn(
             p ) if (p) while (1) break
```

4.9.1.8 nop

```
#define nop( ) asm volatile ("nop"::)
```

4.9.1.9 NULL

```
#define NULL 0
```

4.9.1.10 sti

```
#define sti( ) asm volatile ("sti"::)
```

4.9.1.11 volatile

```
#define volatile __volatile__
```

4.9.2 Typedef Documentation

4.9.2.1 size_t

```
typedef unsigned int size_t
```

4.9.2.2 u16int

```
typedef unsigned short u16int
```

4.9.2.3 u32int

```
typedef unsigned long u32int
```

4.9.2.4 u8int

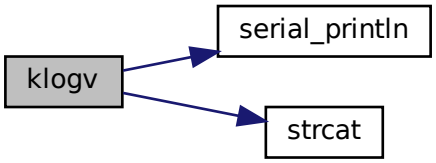
```
typedef unsigned char u8int
```

4.9.3 Function Documentation

4.9.3.1 klogv()

```
void klogv (
            const char * msg )
```

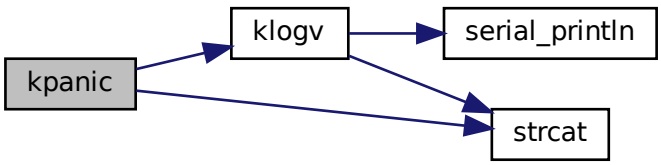
Here is the call graph for this function:



4.9.3.2 kpanic()

```
void kpanic (
            const char * msg )
```

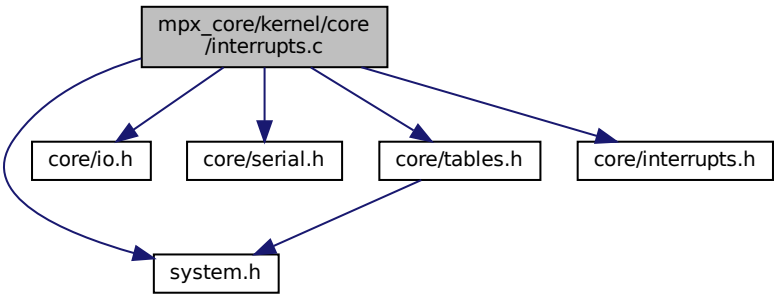
Here is the call graph for this function:



4.10 mpx_core/kernel/core/interrupts.c File Reference

```
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
```

Include dependency graph for interrupts.c:



Macros

- #define PIC1 0x20
- #define PIC2 0xA0
- #define ICW1 0x11
- #define ICW4 0x01
- #define io_wait() asm volatile ("outb $0x80")

Functions

- void divide_error ()
- void debug ()
- void nmi ()
- void breakpoint ()
- void overflow ()
- void bounds ()
- void invalid_op ()
- void device_not_available ()
- void double_fault ()
- void coprocessor_segment ()
- void invalid_tss ()
- void segment_not_present ()
- void stack_segment ()
- void general_protection ()
- void page_fault ()
- void reserved ()
- void coprocessor ()
- void rtc_isr ()

- void sys_call_isr ()
- void serial_io_isr ()
- void isr0 ()
- void do_isr ()
- void init_irq (void)
- void init_pic (void)
- void do_divide_error ()
- void do_debug ()
- void do_nmi ()
- void do_breakpoint ()
- void do_overflow ()
- void do_bounds ()
- void do_invalid_op ()
- void do_device_not_available ()
- void do_double_fault ()
- void do_coprocessor_segment ()
- void do_invalid_tss ()
- void do_segment_not_present ()
- void do_stack_segment ()
- void do_general_protection ()
- void do_page_fault ()
- void do_reserved ()
- void do_coprocessor ()

Variables

- idt_entry idt_entries [256]

4.10.1 Macro Definition Documentation

4.10.1.1 ICW1

```
#define ICW1 0x11
```

4.10.1.2 ICW4

```
#define ICW4 0x01
```

4.10.1.3 io_wait

```
#define io_wait( ) asm volatile ("outb $0x80")
```

4.10.1.4 PIC1

```
#define PIC1 0x20
```

4.10.1.5 PIC2

```
#define PIC2 0xA0
```

4.10.2 Function Documentation

4.10.2.1 bounds()

```
void bounds ( )
```

4.10.2.2 breakpoint()

```
void breakpoint ( )
```

4.10.2.3 coprocessor()

```
void coprocessor ( )
```

4.10.2.4 coprocessor_segment()

```
void coprocessor_segment ( )
```

4.10.2.5 debug()

```
void debug ( )
```

4.10.2.6 device_not_available()

```
void device_not_available ( )
```

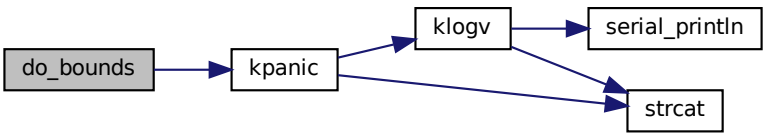
4.10.2.7 divide_error()

```
void divide_error ( )
```

4.10.2.8 do_bounds()

```
void do_bounds ( )
```

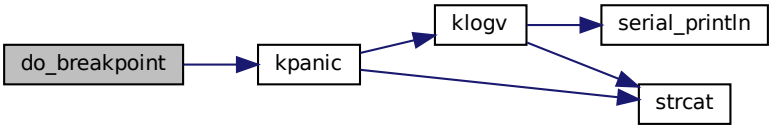
Here is the call graph for this function:



4.10.2.9 do_breakpoint()

```
void do_breakpoint ( )
```

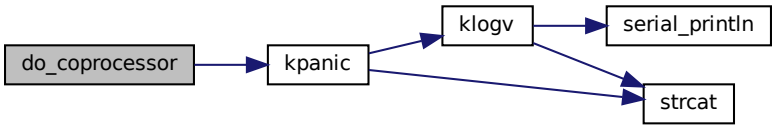
Here is the call graph for this function:



4.10.2.10 do_coprocessor()

```
void do_coprocessor ( )
```

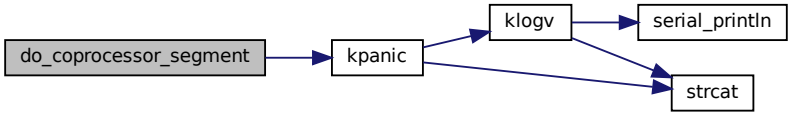
Here is the call graph for this function:



4.10.2.11 do_coprocessor_segment()

```
void do_coprocessor_segment ( )
```

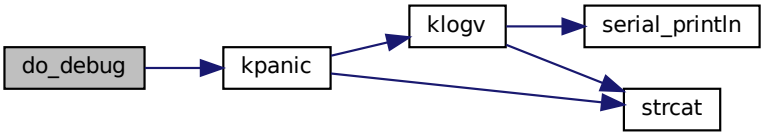
Here is the call graph for this function:



4.10.2.12 do_debug()

```
void do_debug ( )
```

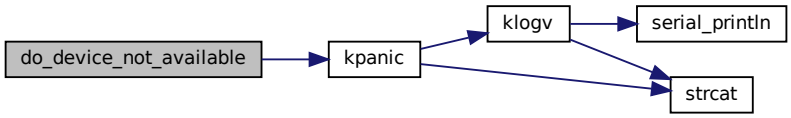
Here is the call graph for this function:



4.10.2.13 do_device_not_available()

```
void do_device_not_available ( )
```

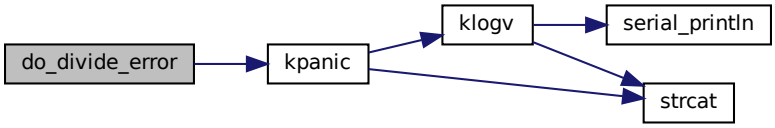
Here is the call graph for this function:



4.10.2.14 do_divide_error()

```
void do_divide_error ( )
```

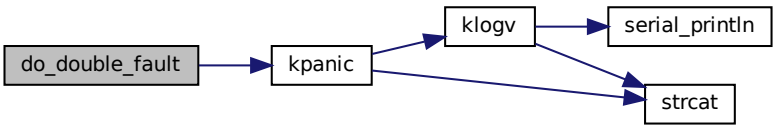
Here is the call graph for this function:



4.10.2.15 do_double_fault()

```
void do_double_fault ( )
```

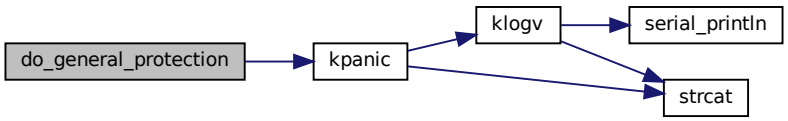
Here is the call graph for this function:



4.10.2.16 do_general_protection()

```
void do_general_protection ( )
```

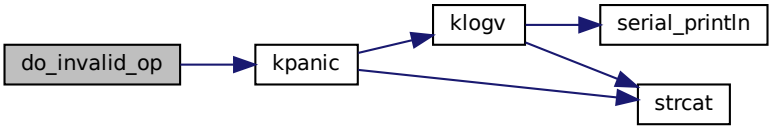
Here is the call graph for this function:



4.10.2.17 do_invalid_op()

```
void do_invalid_op ( )
```

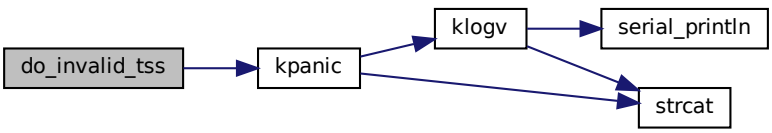
Here is the call graph for this function:



4.10.2.18 do_invalid_tss()

```
void do_invalid_tss ( )
```

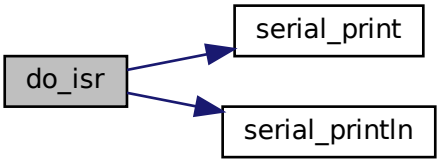
Here is the call graph for this function:



4.10.2.19 do_isr()

```
void do_isr ( )
```

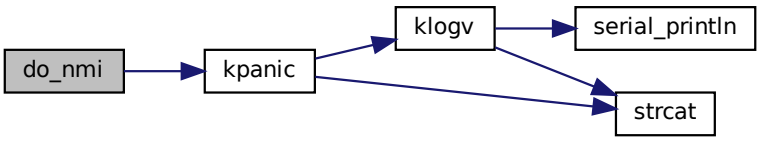
Here is the call graph for this function:



4.10.2.20 do_nmi()

```
void do_nmi ( )
```

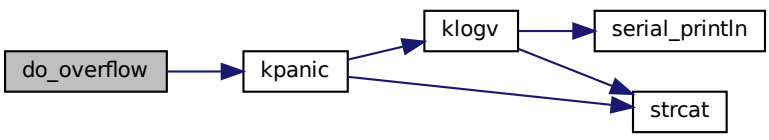
Here is the call graph for this function:



4.10.2.21 do_overflow()

```
void do_overflow ( )
```

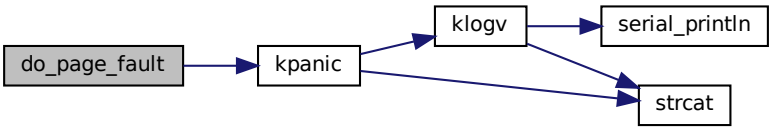
Here is the call graph for this function:



4.10.2.22 do_page_fault()

```
void do_page_fault ( )
```

Here is the call graph for this function:



4.10.2.23 do_reserved()

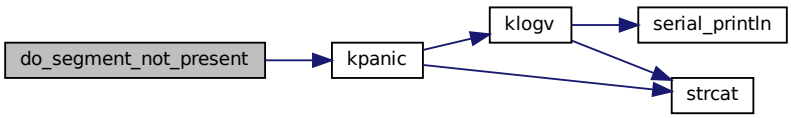
```
void do_reserved ( )
```

Here is the call graph for this function:

4.10.2.24 do_segment_not_present()

```
void do_segment_not_present ( )
```

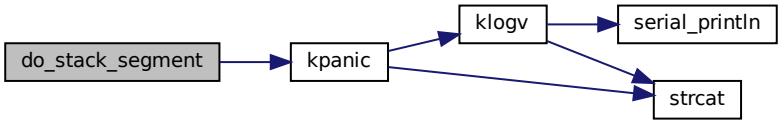
Here is the call graph for this function:



4.10.2.25 do_stack_segment()

```
void do_stack_segment ( )
```

Here is the call graph for this function:



4.10.2.26 double_fault()

```
void double_fault ( )
```

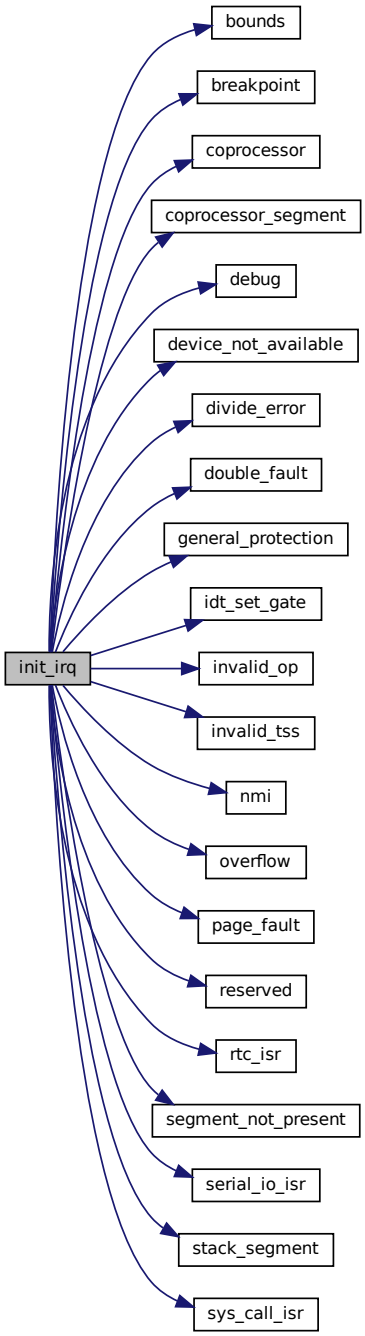
4.10.2.27 general_protection()

```
void general_protection ( )
```

4.10.2.28 init_irq()

```
void init_irq (
            void  )
```

Here is the call graph for this function:



4.10.2.29 init_pic()

```
void init_pic (
            void  )
```

4.10.2.30 invalid_op()

```
void invalid_op ( )
```

4.10.2.31 invalid_tss()

```
void invalid_tss ( )
```

4.10.2.32 isr0()

```
void isr0 ( )
```

4.10.2.33 nmi()

```
void nmi ( )
```

4.10.2.34 overflow()

```
void overflow ( )
```

4.10.2.35 page_fault()

```
void page_fault ( )
```

4.10.2.36 reserved()

```
void reserved ( )
```

4.10.2.37 rtc_isr()

```
void rtc_isr ( )
```

4.10.2.38 segment_not_present()

```
void segment_not_present ( )
```

4.10.2.39 serial_io_isr()

```
void serial_io_isr ( )
```

4.10.2.40 stack_segment()

```
void stack_segment ( )
```

4.10.2.41 sys_call_isr()

```
void sys_call_isr ( )
```

4.10.3 Variable Documentation

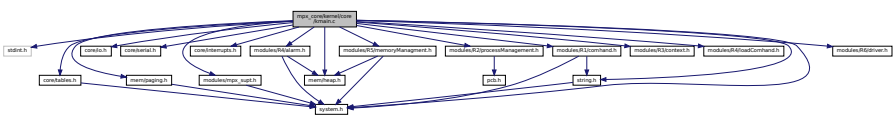
4.10.3.1 idt_entries

```
idt_entry idt_entries[256]  [extern]
```

4.11 mpx_core/kernel/core/kmain.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
#include <mem/heap.h>
#include <mem/paging.h>
#include "modules/mpx_supt.h"
#include "modules/R1/comhand.h"
#include "modules/R2/processManagement.h"
#include "modules/R3/context.h"
#include "modules/R4/loadComhand.h"
#include "modules/R4/alarm.h"
#include "modules/R5/memoryManagment.h"
#include "modules/R6/driver.h"
```

Include dependency graph for kmain.c:



Functions

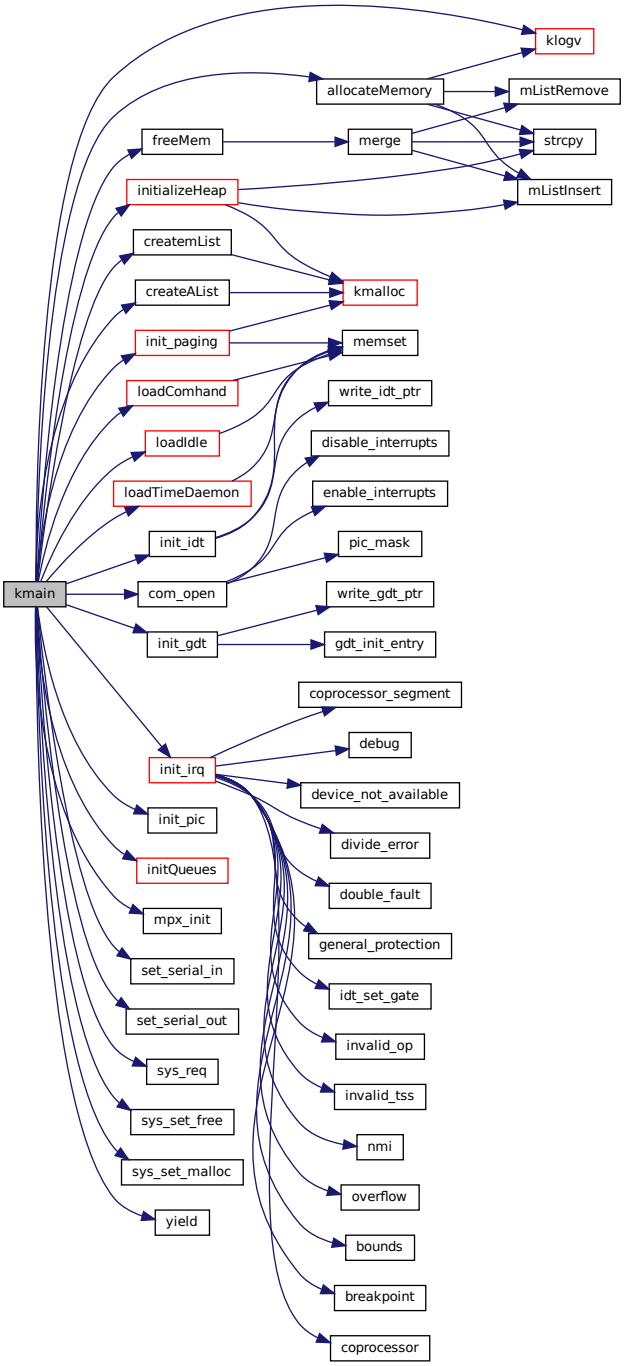
- void kmain (void)

4.11.1 Function Documentation

4.11.1.1 kmain()

```
void kmain (
            void  )
```

Here is the call graph for this function:

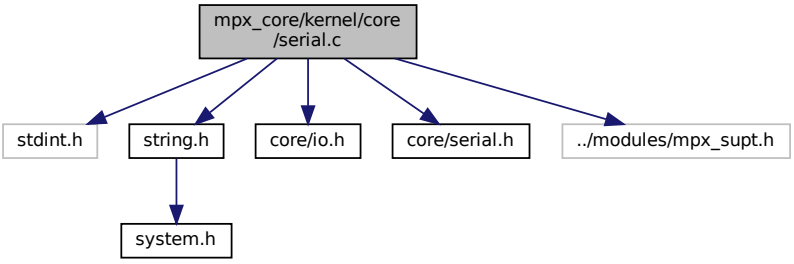


4.12 mpx_core/kernel/core/serial.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <core/io.h>
#include <core/serial.h>
```

```
#include <../modules/mpx_supt.h>
```

Include dependency graph for serial.c:



Macros

- #define NO_ERROR 0
- #define NEWLINE 10
- #define ENTER 13
- #define DELETE 8
- #define DOWN_ARROW 66
- #define ESCAPE 27
- #define RIGHT_ARROW 67
- #define LEFT_ARROW 68
- #define LEFT_BRACKET 91
- #define UP_ARROW 65
- #define BACKSPACE 127
- #define MV_CURSOR_START "\033[1000D"
- #define MV_CURSOR_RIGHT "\033[C"
- #define MV_CURSOR_LEFT "\033[D"
- #define CLEAR_LINE "\033[2K"
- #define PRINT_GREEN "\033[92m"
- #define MAX_SIZE 20
- #define UP "\033[A"
- #define DOWN "\033[B"
- #define max 99

Functions

- int init_serial (int device)
- int serial_println (const char *msg)
- int serial_print (const char *msg)
- int set_serial_out (int device)
- int set_serial_in (int device)
- int * polling (char *buffer, int *count)
- void resetLine ()
- void resetCursor ()
- void letterChar (char *buffer, char *letter)
- void deleteChar (char *buffer)
- void backspaceChar (char *buffer)

- void rightArrowChar ()
- void leftArrowChar ()
- int isUpArrow (char *letter)
- int isDownArrow (char *letter)
- int isRightArrow (char *letter)
- int isLeftArrow (char *letter)
- int isEscape (char *letter)
- int isBackspace (char *letter)
- int isEnter (char *letter)
- int isLetter (char *letter)
- void initializeHistory ()
- int addToHistory (char *command)
- int comEmpty ()
- void getHistory (char *buffer, char *letter)
- char * getCommandUp ()
- char * getCommandDown ()
- int historyFull ()

Variables

- int bufferSize
- int bufferIndex
- int serial_port_out = 0
- int serial_port_in = 0
- char commands [21][31]
- int size
- int index

4.12.1 Macro Definition Documentation

4.12.1.1 BACKSPACE

```
#define BACKSPACE 127
```

4.12.1.2 CLEAR_LINE

```
#define CLEAR_LINE "\033[2K"
```

4.12.1.3 DELETE

```
#define DELETE 8
```

4.12.1.4 DOWN

```
#define DOWN "\033[B"
```

4.12.1.5 DOWN_ARROW

```
#define DOWN_ARROW 66
```

4.12.1.6 ENTER

```
#define ENTER 13
```

4.12.1.7 ESCAPE

```
#define ESCAPE 27
```

4.12.1.8 LEFT_ARROW

```
#define LEFT_ARROW 68
```

4.12.1.9 LEFT_BRACKET

```
#define LEFT_BRACKET 91
```

4.12.1.10 max

```
#define max 99
```

4.12.1.11 MAX_SIZE

```
#define MAX_SIZE 20
```

4.12.1.12 MV_CURSOR_LEFT

```
#define MV_CURSOR_LEFT "\033[D"
```

4.12.1.13 MV_CURSOR_RIGHT

```
#define MV_CURSOR_RIGHT "\033[C"
```

4.12.1.14 MV_CURSOR_START

```
#define MV_CURSOR_START "\033[1000D"
```

4.12.1.15 NEWLINE

```
#define NEWLINE 10
```

4.12.1.16 NO_ERROR

```
#define NO_ERROR 0
```

4.12.1.17 PRINT_GREEN

```
#define PRINT_GREEN "\033[92m"
```

4.12.1.18 RIGHT_ARROW

```
#define RIGHT_ARROW 67
```

4.12.1.19 UP

```
#define UP "\033[A"
```

4.12.1.20 UP_ARROW

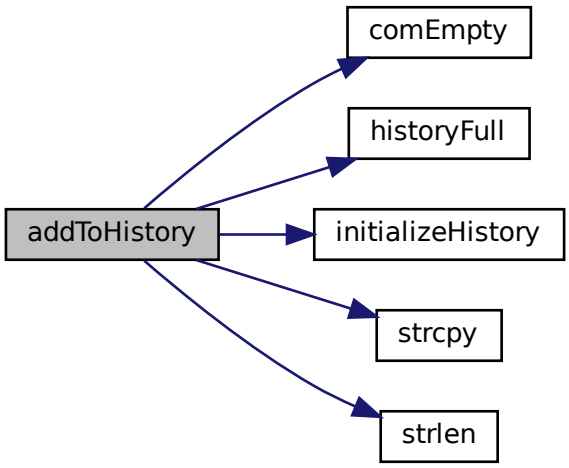
```
#define UP_ARROW 65
```

4.12.2 Function Documentation

4.12.2.1 addToHistory()

```
int addToHistory (
            char * command )
```

Here is the call graph for this function:



4.12.2.2 backspaceChar()

```
void backspaceChar (
            char * buffer )
```

4.12.2.3 Function: backspaceChar

Handles backspace characters. Copies the current buffers content up to the cursor location into a new buffer and concatenates that with everything past the cursor.

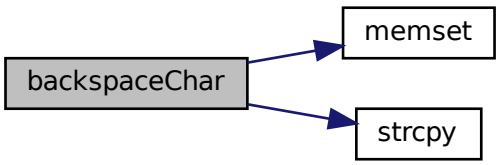
Parameters

∗buffer	the buffer to be changed

Author

Brendan Michael

Here is the call graph for this function:



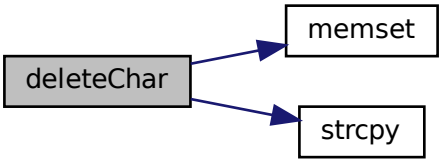
4.12.2.4 comEmpty()

```
int comEmpty ( )
```

4.12.2.5 deleteChar()

```
void deleteChar (
            char * buffer )
```

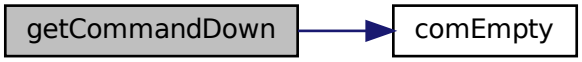
Here is the call graph for this function:



4.12.2.6 getCommandDown()

```
char* getCommandDown ( )
```

Here is the call graph for this function:



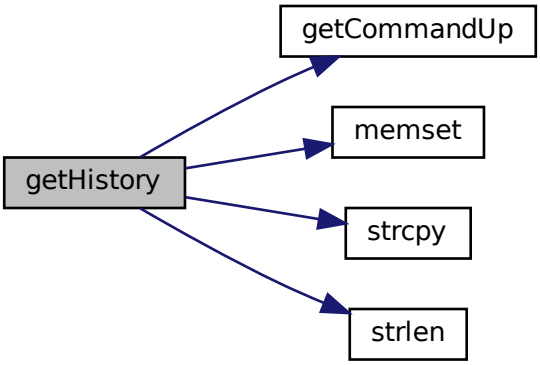
4.12.2.7 getCommandUp()

```
char* getCommandUp ( )
```

4.12.2.8 getHistory()

```
void getHistory (
            char * buffer,
            char * letter )
```

Here is the call graph for this function:



4.12.2.9 historyFull()

```
int historyFull ( )
```

4.12.2.10 init_serial()

```
int init_serial (
            int device )
```

4.12.2.11 initializeHistory()

```
void initializeHistory ( )
```

4.12.2.12 isBackspace()

```
int isBackspace (
            char * letter )
```

4.12.2.13 isDownArrow()

```
int isDownArrow (
            char * letter )
```

4.12.2.14 isEnter()

```
int isEnter (
            char * letter )
```

4.12.2.15 isEscape()

```
int isEscape (
            char * letter )
```

4.12.2.16 isLeftArrow()

```
int isLeftArrow (
            char * letter )
```

4.12.2.17 isLetter()

```
int isLetter (
            char * letter )
```

4.12.2.18 isRightArrow()

```
int isRightArrow (
            char * letter )
```

4.12.2.19 isUpArrow()

```
int isUpArrow (
            char * letter )
```

4.12.2.20 leftArrowChar()

```
void leftArrowChar ( )
```

4.12.2.21 Function: leftArrowChar

Moves the cursor one space to the left and decrements the buffer index.

Parameters

∗buffer	the buffer to be changed

Author

Brendan Michael

Here is the call graph for this function:

4.12.2.22 letterChar()

```
void letterChar (
            char * buffer,
            char * letter )
```

4.12.2.23 polling()

```
int* polling (
            char * buffer,
            int * count )
```

4.12.2.24 Function: polling

Fills the buffer with keyboard input and returns to command handler when enter key is pressed. Also handles special characters

Parameters

∗buffer	String buffer to be filled
∗count	size of the buffer

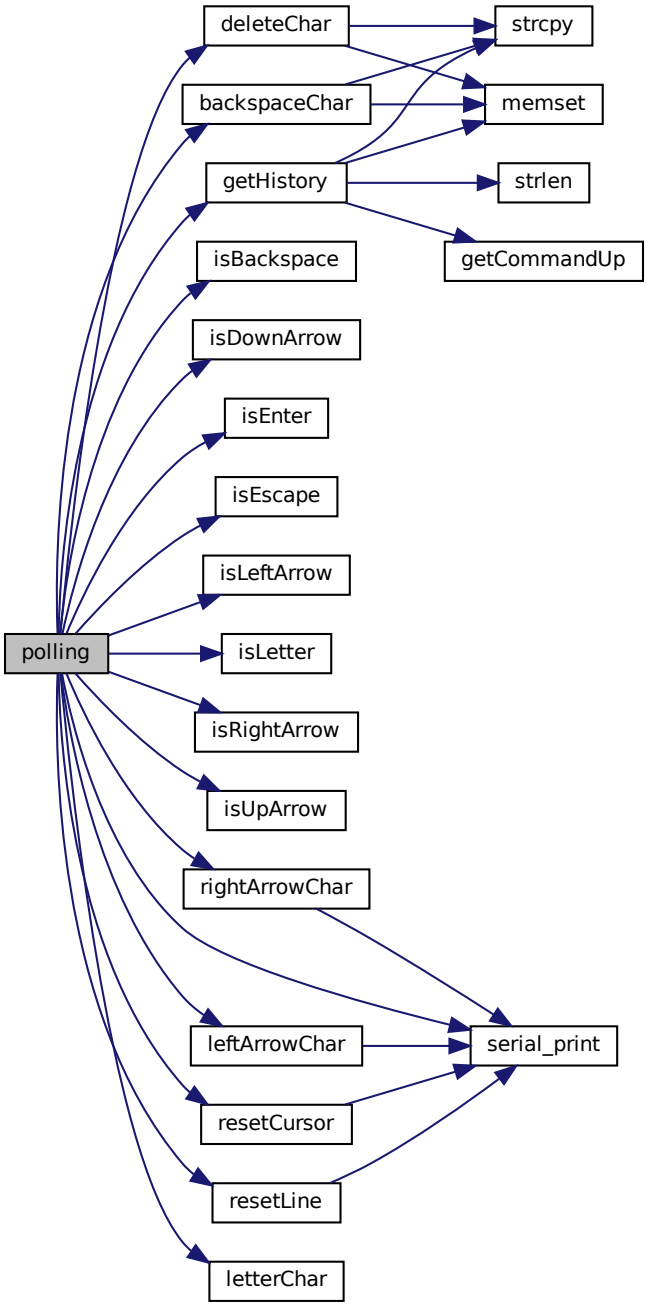
Returns

the buffer size

Author

Brendan Michael

Here is the call graph for this function:



4.12.2.25 resetCursor()

```
void resetCursor ( )
```

4.12.2.26 Function: resetCursor

Using escape sequences, moves the cursor from the start of the line up to the current location in the buffer

Author

Brendan Michael

Here is the call graph for this function:

4.12.2.27 resetLine()

```
void resetLine ( )
```

4.12.2.28 Function: resetLine

Using escape sequences, moves the cursor to the beginning of the line, clears it, and changes the color to green.

Author

Brendan Michael

Here is the call graph for this function:

4.12.2.29 rightArrowChar()

```
void rightArrowChar ( )
```

4.12.2.30 Function: righArrowChar

Moves the cursor one space to the right and increments the buffer index.

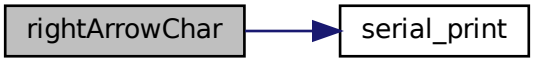
Parameters

∗buffer	the buffer to be changed

Author

Brendan Michael

Here is the call graph for this function:



4.12.2.31 serial_print()

```
int serial_print (
            const char * msg )
```

4.12.2.32 serial_println()

```
int serial_println (
            const char * msg )
```

4.12.2.33 set_serial_in()

```
int set_serial_in (
            int device )
```

4.12.2.34 set_serial_out()

```
int set_serial_out (
            int device )
```

4.12.3 Variable Documentation

4.12.3.1 bufferIndex

```
int bufferIndex
```

4.12.3.2 bufferSize

```
int bufferSize
```

4.12.3.3 commands

```
char commands[21][31]
```

4.12.3.4 index

```
int index
```

4.12.3.5 serial_port_in

```
int serial_port_in = 0
```

4.12.3.6 serial_port_out

```
int serial_port_out = 0
```

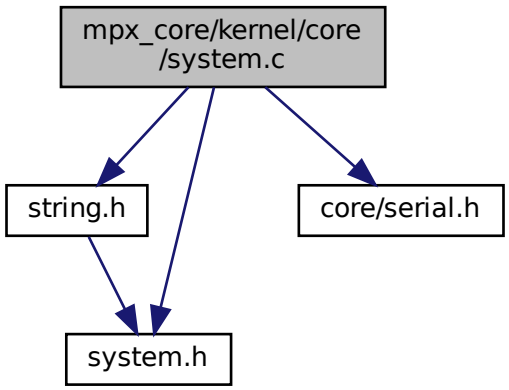
4.12.3.7 size

```
int size
```

4.13 mpx_core/kernel/core/system.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
```

Include dependency graph for system.c:



Functions

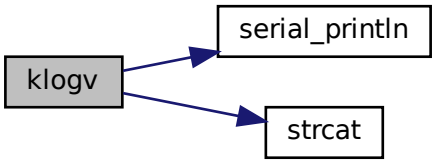
- void klogv (const char *msg)
- void kpanic (const char *msg)

4.13.1 Function Documentation

4.13.1.1 klogv()

```
void klogv (
            const char * msg )
```

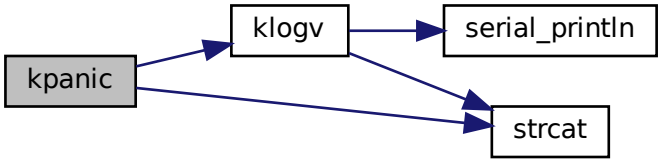
Here is the call graph for this function:



4.13.1.2 kpanic()

```
void kpanic (
            const char * msg )
```

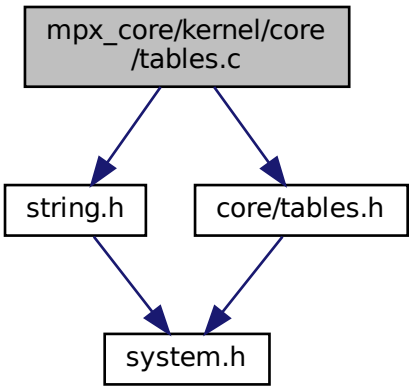
Here is the call graph for this function:



4.14 mpx_core/kernel/core/tables.c File Reference

```
#include <string.h>
#include <core/tables.h>
```

Include dependency graph for tables.c:



Functions

- void write_gdt_ptr (u32int, size_t)
- void write_idt_ptr (u32int)
- void idt_set_gate (u8int idx, u32int base, u16int sel, u8int flags)
- void init_idt ()
- void gdt_init_entry (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void init_gdt ()

Variables

- gdt_descriptor gdt_ptr
- gdt_entry gdt_entries [5]
- idt_descriptor idt_ptr
- idt_entry idt_entries [256]

4.14.1 Function Documentation

4.14.1.1 gdt_init_entry()

```
void gdt_init_entry (
             int idx,
             u32int base,
             u32int limit,
             u8int access,
             u8int flags )
```

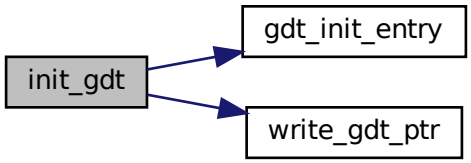
4.14.1.2 idt_set_gate()

```
void idt_set_gate (
             u8int idx,
             u32int base,
             u16int sel,
             u8int flags )
```

4.14.1.3 init_gdt()

```
void init_gdt ( )
```

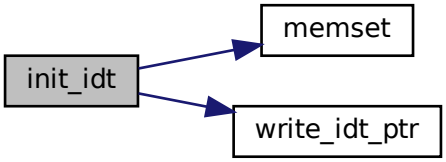
Here is the call graph for this function:



4.14.1.4 init_idt()

```
void init_idt ( )
```

Here is the call graph for this function:



4.14.1.5 write_gdt_ptr()

```
void write_gdt_ptr (
             u32int ,
             size_t  )
```

4.14.1.6 write_idt_ptr()

```
void write_idt_ptr (
             u32int  )
```

4.14.2 Variable Documentation

4.14.2.1 gdt_entries

```
gdt_entry gdt_entries[5]
```

4.14.2.2 gdt_ptr

```
gdt_descriptor gdt_ptr
```

4.14.2.3 idt_entries

```
idt_entry idt_entries[256]
```

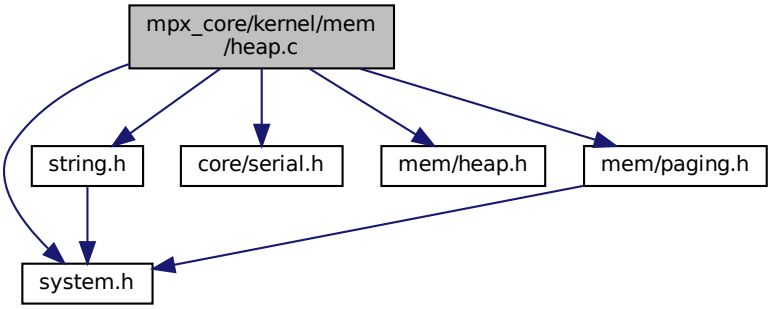
4.14.2.4 idt_ptr

```
idt_descriptor idt_ptr
```

4.15 mpx_core/kernel/mem/heap.c File Reference

```
#include <system.h>
#include <string.h>
#include <core/serial.h>
#include <mem/heap.h>
#include <mem/paging.h>
```

Include dependency graph for heap.c:



Functions

- u32int _kmalloc (u32int size, int page_align, u32int *phys_addr)
- u32int kmalloc (u32int size)
- u32int alloc (u32int size, heap *h, int align)
- heap * make_heap (u32int base, u32int max, u32int min)

Variables

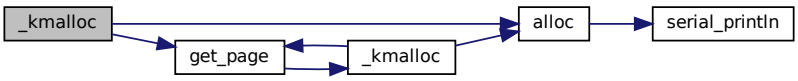
- heap * kheap = 0
- heap * curr_heap = 0
- page_dir * kdir
- void * end
- void _end
- void __end
- u32int phys_alloc_addr = (u32int)&end

4.15.1 Function Documentation

4.15.1.1 _kmalloc()

```
u32int _kmalloc (
            u32int size,
            int page_align,
            u32int * phys_addr )
```

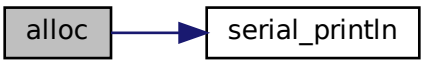
Here is the call graph for this function:



4.15.1.2 alloc()

```
u32int alloc (
            u32int size,
            heap * h,
            int align )
```

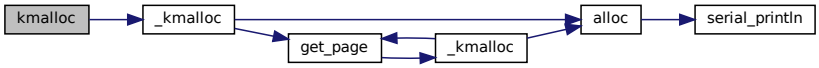
Here is the call graph for this function:



4.15.1.3 kmalloc()

```
u32int kmalloc (
            u32int size )
```

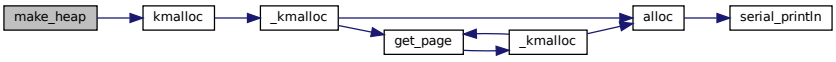
Here is the call graph for this function:



4.15.1.4 make_heap()

```
heap* make_heap (
            u32int base,
            u32int max,
            u32int min )
```

Here is the call graph for this function:



4.15.2 Variable Documentation

4.15.2.1 __end

```
void __end
```

4.15.2.2 _end

```
void _end
```

4.15.2.3 curr_heap

```
heap* curr_heap = 0
```

4.15.2.4 end

```
void* end  [extern]
```

4.15.2.5 kdir

```
page_dir* kdir  [extern]
```

4.15.2.6 kheap

```
heap* kheap = 0
```

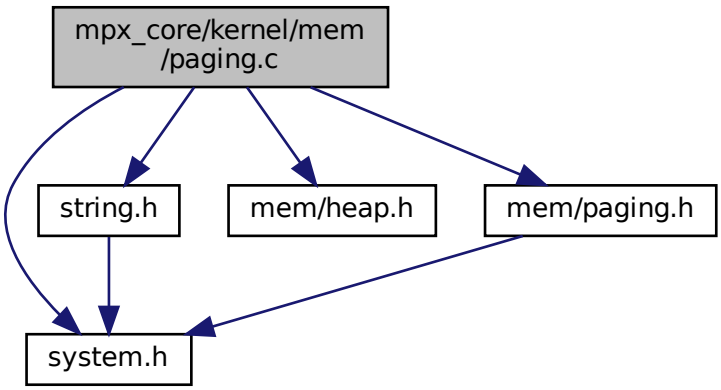
4.15.2.7 phys_alloc_addr

```
u32int phys_alloc_addr = (u32int)&end
```

4.16 mpx_core/kernel/mem/paging.c File Reference

```
#include <system.h>
#include <string.h>
#include "mem/heap.h"
#include "mem/paging.h"
```

Include dependency graph for paging.c:



Functions

- void set_bit (u32int addr)
- void clear_bit (u32int addr)
- u32int get_bit (u32int addr)
- u32int find_free ()
- page_entry * get_page (u32int addr, page_dir *dir, int make_table)
- void init_paging ()
- void load_page_dir (page_dir *new_dir)
- void new_frame (page_entry *page)

Variables

- u32int mem_size = 0x4000000
- u32int page_size = 0x1000
- u32int nframes
- u32int * frames
- page_dir * kdir = 0
- page_dir * cdir = 0
- u32int phys_alloc_addr
- heap * kheap

4.16.1 Function Documentation

4.16.1.1 clear_bit()

```
void clear_bit (
            u32int addr )
```

4.16.1.2 find_free()

```
u32int find_free ( )
```

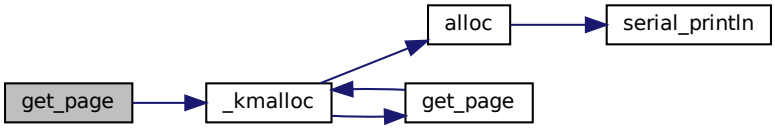
4.16.1.3 get_bit()

```
u32int get_bit (
            u32int addr )
```

4.16.1.4 get_page()

```
page_entry* get_page (
            u32int addr,
            page_dir * dir,
            int make_table )
```

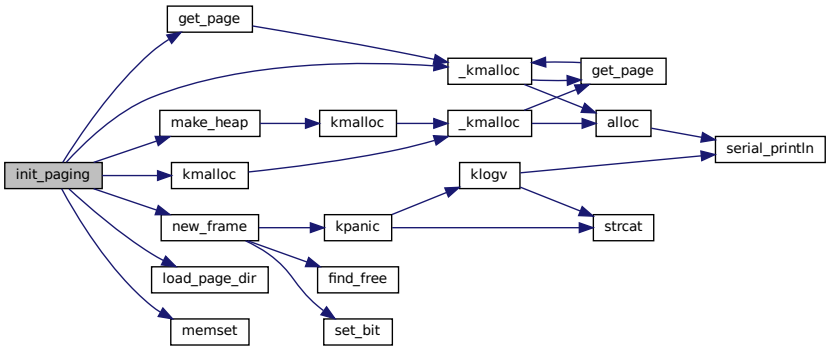
Here is the call graph for this function:



4.16.1.5 init_paging()

```
void init_paging ( )
```

Here is the call graph for this function:



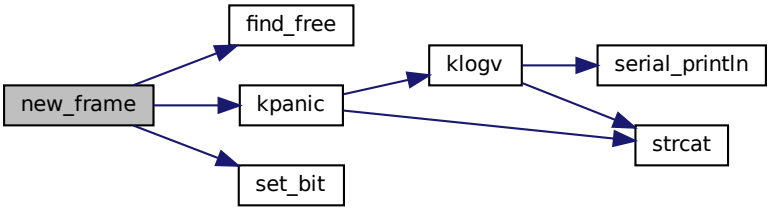
4.16.1.6 load_page_dir()

```
void load_page_dir (
            page_dir * new_dir )
```

4.16.1.7 new_frame()

```
void new_frame (
            page_entry * page )
```

Here is the call graph for this function:



4.16.1.8 set_bit()

```
void set_bit (
            u32int addr )
```

4.16.2 Variable Documentation

4.16.2.1 cdir

```
page_dir* cdir = 0
```

4.16.2.2 frames

```
u32int* frames
```

4.16.2.3 kdir

```
page_dir* kdir = 0
```

4.16.2.4 kheap

```
heap* kheap  [extern]
```

4.16.2.5 mem_size

```
u32int mem_size = 0x4000000
```

4.16.2.6 nframes

```
u32int nframes
```

4.16.2.7 page_size

```
u32int page_size = 0x1000
```

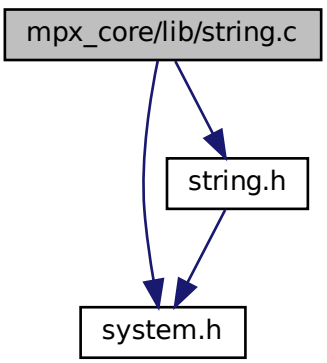
4.16.2.8 phys_alloc_addr

```
u32int phys_alloc_addr  [extern]
```

4.17 mpx_core/lib/string.c File Reference

```
#include <system.h>
#include <string.h>
```

Include dependency graph for string.c:



Functions

- int strlen (const char *s)
- char * strcpy (char *s1, const char *s2)
- int atoi (const char *s)
- int strcmp (const char *s1, const char *s2)
- char * strcat (char *s1, const char *s2)
- int isspace (const char *c)
- void * memset (void *s, int c, size_t n)
- char * strtok (char *s1, const char *s2)
- int strcasecmp (const char *s1, const char *s2)
- char toupper (char x)
- int abs (int number)
- void swap (char *x, char *y)
- char * reverse (char *buffer, int i, int j)
- char * itoa (int value, char *buffer, int base)

4.17.1 Function Documentation

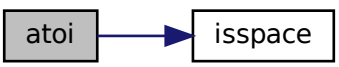
4.17.1.1 abs()

```
int abs (
            int number )
```

4.17.1.2 atoi()

```
int atoi (
            const char * s )
```

Here is the call graph for this function:



4.17.1.3 isspace()

```
int isspace (
            const char * c )
```

4.17.1.4 itoa()

```
char* itoa (
            int value,
            char * buffer,
            int base )
```

4.17.1.5 Function: itoa

Iterative function to implement itoa() function in C geeksforgeeks.org/implement-itoa/ for reference. Changed a little to make own.

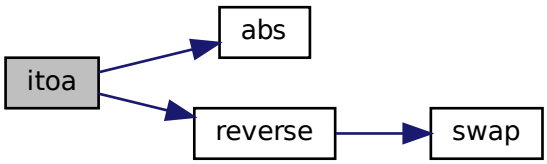
Parameters

value	integer taken as input
base	integer of the desired base to convert.
buffer	character buffer where the written output is being passed to.

Author

Bryce Williams

Here is the call graph for this function:



4.17.1.6 memset()

```
void* memset (
            void * s,
            int c,
            size_t n )
```

4.17.1.7 reverse()

```
char* reverse (
            char * buffer,
            int i,
            int j )
```

Here is the call graph for this function:

4.17.1.8 strcasecmp()

```
int strcasecmp (
            const char * s1,
            const char * s2 )
```

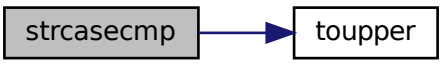
4.17.1.9 Function: strcasecmp

Compares two strings while being case insentsitive.

Author

Bryce Williams

Here is the call graph for this function:



4.17.1.10 strcat()

```
char* strcat (
            char * s1,
            const char * s2 )
```

4.17.1.11 strcmp()

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

4.17.1.12 strcpy()

```
char* strcpy (
            char * s1,
            const char * s2 )
```

4.17.1.13 strlen()

```
int strlen (
            const char * s )
```

4.17.1.14 strtok()

```
char* strtok (
            char * s1,
            const char * s2 )
```

4.17.1.15 swap()

```
void swap (
            char * x,
            char * y )  [inline]
```

4.17.1.16 toupper()

```
char toupper (
            char x )
```

4.17.1.17 Function: toupper

Converts character to uppercase. ex: 'a' -> 'A'

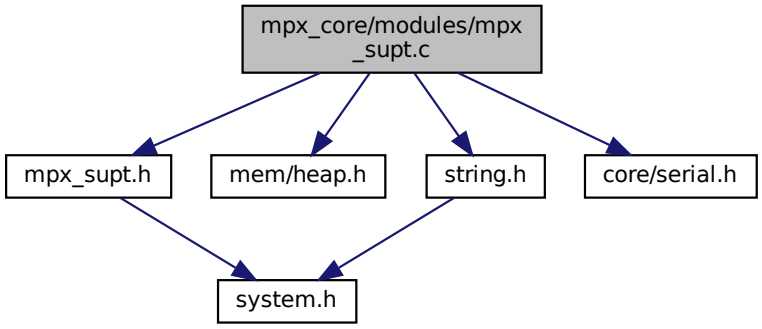
Author

Bryce Williams

4.18 mpx_core/modules/mpx_supt.c File Reference

```
#include "mpx_supt.h"
#include <mem/heap.h>
#include <string.h>
#include <core/serial.h>
```

Include dependency graph for mpx_supt.c:



Functions

- int is_io_module_active ()
- int sys_req (int op_code, int device_id, char *buffer_ptr, int *count_ptr)
- void mpx_init (int cur_mod)
- void sys_set_malloc (u32int(*func)(u32int))
- void sys_set_free (int(*func)(void *))
- void * sys_alloc_mem (u32int size)
- int sys_free_mem (void *ptr)
- void idle ()
- void printMessage (char *message)
- void printlnMessage (char *message)

Variables

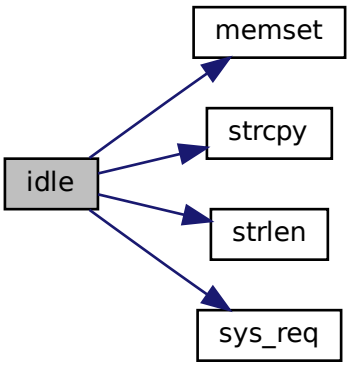
- param params
- int current_module = -1
- u32int(* student_malloc)(u32int)
- int(* student_free)(void *)

4.18.1 Function Documentation

4.18.1.1 idle()

```
void idle ( )
```

Here is the call graph for this function:



4.18.1.2 is_io_module_active()

```
int is_io_module_active ( )
```

4.18.1.3 mpx_init()

```
void mpx_init (
            int cur_mod )
```

4.18.1.4 printlnMessage()

```
void printlnMessage (
            char * message )
```

4.18.1.5 Function: printMessage

Calls sysreq to display a user entered message while a starting a newline.

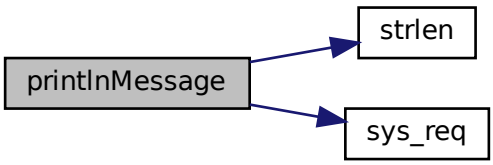
Parameters

message	character pointer pointing to text inside message.

Author

Bryce Williams

Here is the call graph for this function:



4.18.1.6 printMessage()

```
void printMessage (
            char * message )
```

4.18.1.7 Function: printMessage

Calls sysreq to display a user entered message without a starting a newline.

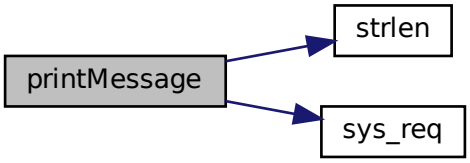
Parameters

message	character pointer pointing to text inside message.

Author

Bryce Williams

Here is the call graph for this function:



4.18.1.8 sys_alloc_mem()

```
void* sys_alloc_mem (
            u32int size )
```

4.18.1.9 sys_free_mem()

```
int sys_free_mem (
            void * ptr )
```

4.18.1.10 sys_req()

```
int sys_req (
            int op_code,
            int device_id,
            char * buffer_ptr,
            int * count_ptr )
```

4.18.1.11 sys_set_free()

```
void sys_set_free (
            int(*)(void *) func )
```

4.18.1.12 sys_set_malloc()

```
void sys_set_malloc (
            u32int(*)(u32int) func )
```

4.18.2 Variable Documentation

4.18.2.1 current_module

```
int current_module = -1
```

4.18.2.2 params

```
param params
```

4.18.2.3 student_free

```
int(* student_free) (void *)
```

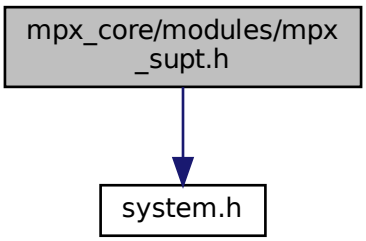
4.18.2.4 student_malloc

```
u32int(* student_malloc) (u32int)
```

4.19 mpx_core/modules/mpx_supt.h File Reference

```
#include <system.h>
```

Include dependency graph for mpx_supt.h:



This graph shows which files directly or indirectly include this file:

Data Structures

- struct param

Macros

- #define EXIT 0
- #define IDLE 1
- #define READ 2
- #define WRITE 3
- #define INVALID_OPERATION 4
- #define TRUE 1
- #define FALSE 0
- #define MODULE_R1 0
- #define MODULE_R2 1
- #define MODULE_R3 2
- #define MODULE_R4 4
- #define MODULE_R5 8
- #define MODULE_F 9
- #define IO_MODULE 10
- #define MEM_MODULE 11
- #define INVALID_BUFFER 1000
- #define INVALID_COUNT 2000
- #define DEFAULT_DEVICE 111
- #define COM_PORT 222
- #define NO_ERROR 0

Functions

- int sys_req (int op_code, int device_id, char *buffer_ptr, int *count_ptr)
- void mpx_init (int cur_mod)
- void sys_set_malloc (u32int(*func)(u32int))
- void sys_set_free (int(*func)(void *))
- void * sys_alloc_mem (u32int size)
- int sys_free_mem (void *ptr)
- void idle ()
- void printMessage (char *message)
- void printlnMessage (char *message)
- int is_io_module_active ()

4.19.1 Macro Definition Documentation

4.19.1.1 COM_PORT

```
#define COM_PORT 222
```

4.19.1.2 DEFAULT_DEVICE

```
#define DEFAULT_DEVICE 111
```

4.19.1.3 EXIT

```
#define EXIT 0
```

4.19.1.4 FALSE

```
#define FALSE 0
```

4.19.1.5 IDLE

```
#define IDLE 1
```

4.19.1.6 INVALID_BUFFER

```
#define INVALID_BUFFER 1000
```

4.19.1.7 INVALID_COUNT

```
#define INVALID_COUNT 2000
```

4.19.1.8 INVALID_OPERATION

```
#define INVALID_OPERATION 4
```

4.19.1.9 IO_MODULE

```
#define IO_MODULE 10
```

4.19.1.10 MEM_MODULE

```
#define MEM_MODULE 11
```

4.19.1.11 MODULE_F

```
#define MODULE_F 9
```

4.19.1.12 MODULE_R1

```
#define MODULE_R1 0
```

4.19.1.13 MODULE_R2

```
#define MODULE_R2 1
```

4.19.1.14 MODULE_R3

```
#define MODULE_R3 2
```

4.19.1.15 MODULE_R4

```
#define MODULE_R4 4
```

4.19.1.16 MODULE_R5

```
#define MODULE_R5 8
```

4.19.1.17 NO_ERROR

```
#define NO_ERROR 0
```

4.19.1.18 READ

```
#define READ 2
```

4.19.1.19 TRUE

```
#define TRUE 1
```

4.19.1.20 WRITE

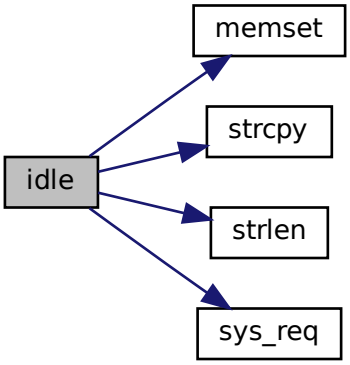
```
#define WRITE 3
```

4.19.2 Function Documentation

4.19.2.1 idle()

```
void idle ( )
```

Here is the call graph for this function:



4.19.2.2 is_io_module_active()

```
int is_io_module_active ( )
```

4.19.2.3 mpx_init()

```
void mpx_init (
            int cur_mod )
```

4.19.2.4 printlnMessage()

```
void printlnMessage (
            char * message )
```

4.19.2.5 Function: printMessage

Calls sysreq to display a user entered message while a starting a newline.

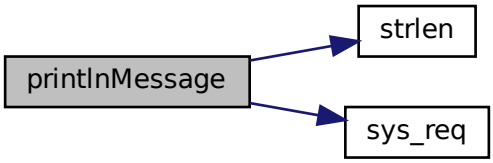
Parameters

message	character pointer pointing to text inside message.

Author

Bryce Williams

Here is the call graph for this function:



4.19.2.6 printMessage()

```
void printMessage (
            char * message )
```

4.19.2.7 Function: printMessage

Calls sysreq to display a user entered message without a starting a newline.

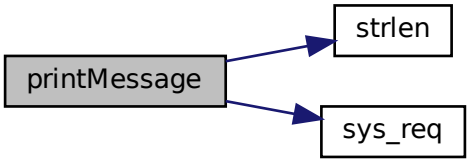
Parameters

message	character pointer pointing to text inside message.

Author

Bryce Williams

Here is the call graph for this function:



4.19.2.8 sys_alloc_mem()

```
void* sys_alloc_mem (
            u32int size )
```

4.19.2.9 sys_free_mem()

```
int sys_free_mem (
            void * ptr )
```

4.19.2.10 sys_req()

```
int sys_req (
            int op_code,
            int device_id,
            char * buffer_ptr,
            int * count_ptr )
```

4.19.2.11 sys_set_free()

```
void sys_set_free (
            int(*)(void *) func )
```

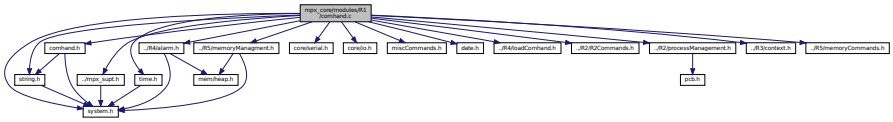
4.19.2.12 sys_set_malloc()

```
void sys_set_malloc (
            u32int(*)(u32int) func )
```

4.20 mpx_core/modules/R1/comhand.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include "../mpx_supt.h"
#include <core/io.h>
#include "comhand.h"
#include "miscCommands.h"
#include "date.h"
#include "time.h"
#include "../R4/alarm.h"
#include "../R4/loadComhand.h"
#include "../R2/R2Commands.h"
#include "../R2/processManagement.h"
#include "../R3/context.h"
#include "../R5/memoryCommands.h"
#include "../R5/memoryManagment.h"
```

Include dependency graph for comhand.c:



Macros

- #define RED "\033[91m"
- #define GREEN "\033[92m"
- #define WHITE "\033[97m"
- #define UP "\033[A"
- #define DOWN "\033[B"

Functions

- int comhand ()
- void noSuchCommand ()
- void poll ()
- void resetBuffer ()
- void printRed ()
- void printGreen ()
- void printWhite ()

Variables

- int introSize = 32
- char intro [ ] = "\nType help for commands list"
- int cmdBufferSize = 99
- char cmdBuffer [100]

4.20.1 Macro Definition Documentation

4.20.1.1 DOWN

```
#define DOWN "\033[B"
```

4.20.1.2 GREEN

```
#define GREEN "\033[92m"
```

4.20.1.3 RED

```
#define RED "\033[91m"
```

4.20.1.4 UP

```
#define UP "\033[A"
```

4.20.1.5 WHITE

```
#define WHITE "\033[97m"
```

4.20.2 Function Documentation

4.20.2.1 comhand()

```
int comhand ( )
```

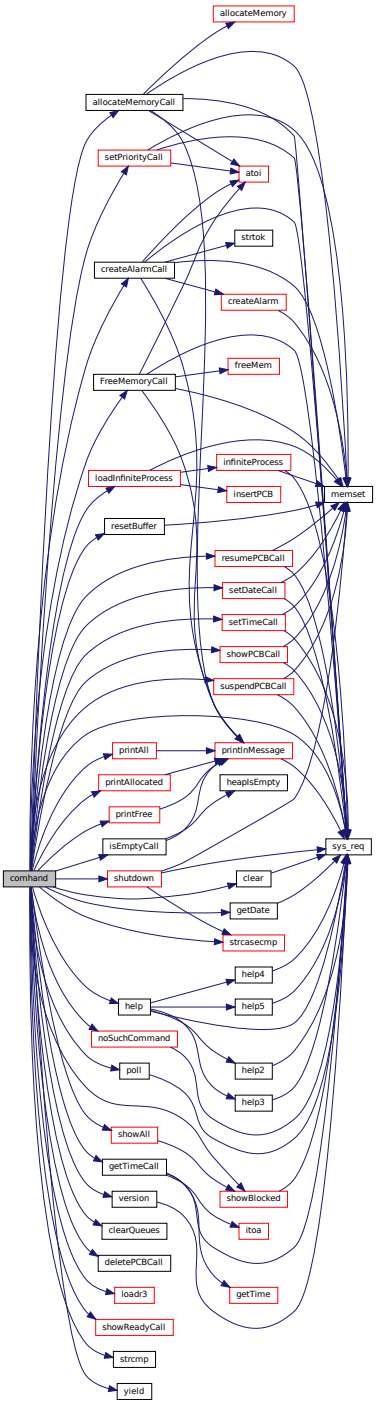
4.20.2.2 Function: comhand

Executes commands based on input gathered from calling polling. Displays an error message if the entered command does not exist.

Authors

Selim Demircan, Brendan Michael Bryce Williams, Farhan Shahbaz

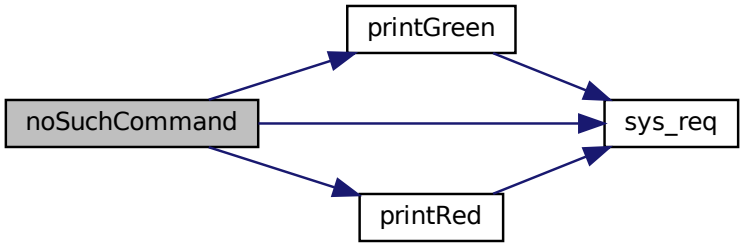
Here is the call graph for this function:



4.20.2.3 noSuchCommand()

```
void noSuchCommand ( )
```

Here is the call graph for this function:



4.20.2.4 poll()

```
void poll ( )
```

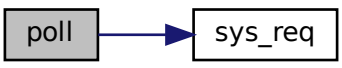
4.20.2.5 Function: poll

Calls polling from within the command handler.

Author

Brendan Michael

Here is the call graph for this function:



4.20.2.6 printGreen()

```
void printGreen ( )
```

Here is the call graph for this function:

4.20.2.7 printRed()

```
void printRed ( )
```

Here is the call graph for this function:

4.20.2.8 printWhite()

```
void printWhite ( )
```

Here is the call graph for this function:

4.20.2.9 resetBuffer()

```
void resetBuffer ( )
```

Here is the call graph for this function:

4.20.3 Variable Documentation

4.20.3.1 cmdBuffer

```
char cmdBuffer[100]
```

4.20.3.2 cmdBufferSize

```
int cmdBufferSize = 99
```

4.20.3.3 intro

```
char intro[] = "\nType help for commands list"
```

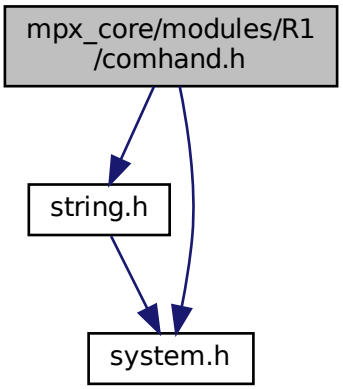
4.20.3.4 introSize

```
int introSize = 32
```

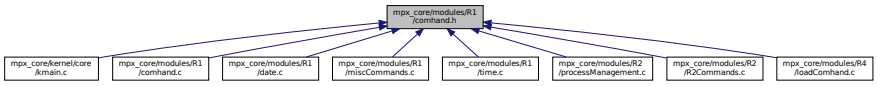
4.21 mpx_core/modules/R1/comhand.h File Reference

```
#include <string.h>
#include <system.h>
```

Include dependency graph for comhand.h:



This graph shows which files directly or indirectly include this file:



Functions

- int comhand ()
- void poll ()
- void deletePCBCall ()
- void noSuchCommand ()
- void createPCBC ()
- void printRed ()
- void printWhite ()
- void printGreen ()
- void resetBuffer ()

4.21.1 Function Documentation

4.21.1.1 comhand()

```
int comhand ( )
```

4.21.1.2 Function: comhand

Executes commands based on input gathered from calling polling. Displays an error message if the entered command does not exist.

Authors

Brendan Michael, Selim Demircan, Bryce Williams, Farhan Shahbaz

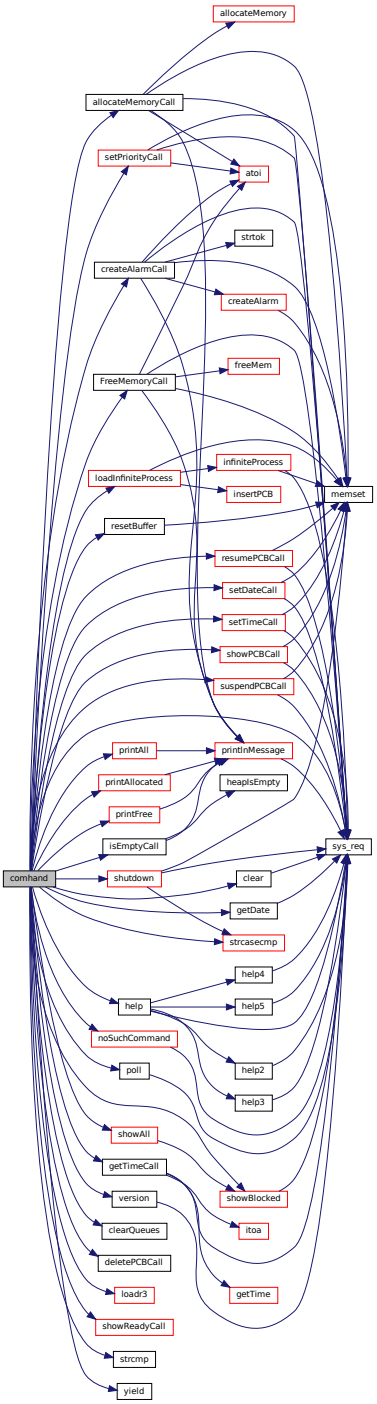
4.21.1.3 Function: comhand

Executes commands based on input gathered from calling polling. Displays an error message if the entered command does not exist.

Authors

Selim Demircan, Brendan Michael Bryce Williams, Farhan Shahbaz

Here is the call graph for this function:



4.21.1.4 createPCBC()

```
void createPCBC ( )
```

4.21.1.5 deletePCBCall()

```
void deletePCBCall ( )
```

4.21.1.6 Function: noSuchCommand

Displays an error message when an entered command is invalid.

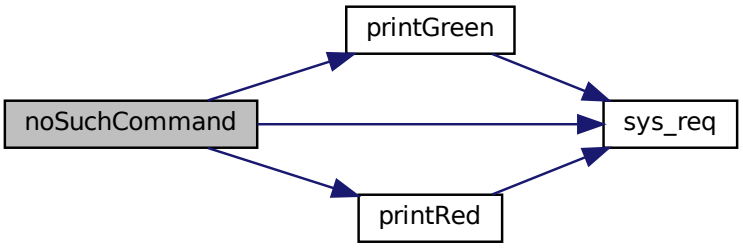
Author

Brendan Michael

4.21.1.7 noSuchCommand()

```
void noSuchCommand ( )
```

Here is the call graph for this function:



4.21.1.8 poll()

```
void poll ( )
```

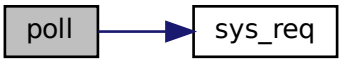
4.21.1.9 Function: poll

Calls polling from within the command handler.

Author

Brendan Michael

Here is the call graph for this function:



4.21.1.10 printGreen()

```
void printGreen ( )
```

Here is the call graph for this function:

4.21.1.11 printRed()

```
void printRed ( )
```

Here is the call graph for this function:

4.21.1.12 printWhite()

```
void printWhite ( )
```

Here is the call graph for this function:

4.21.1.13 resetBuffer()

```
void resetBuffer ( )
```

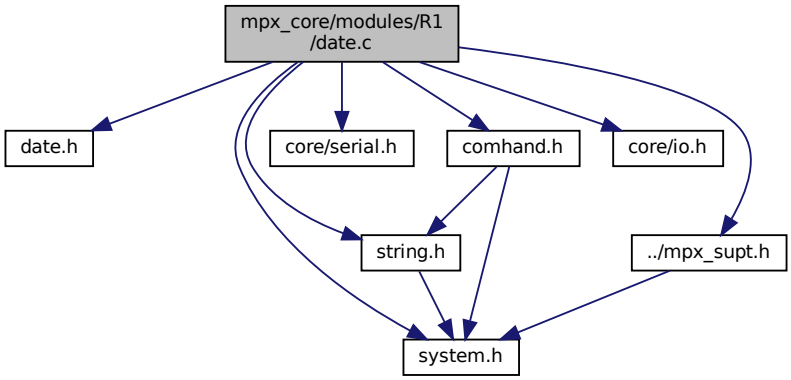
Here is the call graph for this function:

4.22 mpx_core/modules/R1/date.c File Reference

```
#include "date.h"
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include "../mpx_supt.h"
#include <core/io.h>
```

```
#include "comhand.h"
```

Include dependency graph for date.c:



Functions

- void setDate (int day, int month, int year)
- void getDate ()
- void parseDate (char ∗date, int ∗m, int ∗d, int ∗y)
- void setDateCall ()

4.22.1 Function Documentation

4.22.1.1 getDate()

```
void getDate ( )
```

4.22.1.2 Function: getDate

Displays the current date in the format mm/dd/yyy Default time zone is UTC.

Author

Farhan Shahbaz

Here is the call graph for this function:

4.22.1.3 parseDate()

```
void parseDate (
             char * date,
             int * m,
             int * d,
             int * y )
```

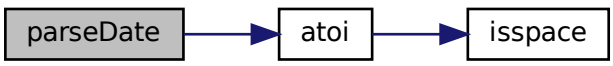
4.22.1.4 Function: parseDate

Parser is used to get and seperate the digit values from the '/' in the date. Makes sure it is in porper form.

Author

Farhan Shahbaz

Here is the call graph for this function:



4.22.1.5 setDate()

```
void setDate (
             int day,
             int month,
             int year )
```

4.22.1.6 Function: setDate

Prompts the user for what date they want to set. The function accesses the day, month, and year registers. Needs to be in the dd/mm/yyyy format.

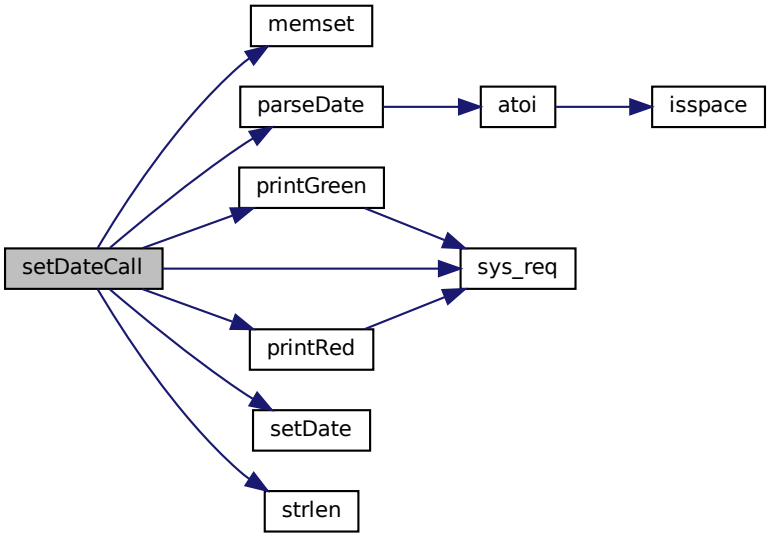
Author

Farhan Shahbaz

4.22.1.7 setDateCall()

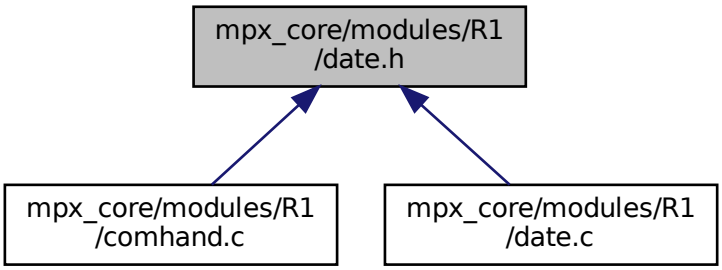
```
void setDateCall ( )
```

Here is the call graph for this function:



4.23 mpx_core/modules/R1/date.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void setDate (int day, int month, int year)
- void getDate ()
- void parseDate (char ∗date, int ∗m, int ∗d, int ∗y)
- void setDateCall ()

4.23.1 Function Documentation

4.23.1.1 getDate()

```
void getDate ( )
```

4.23.1.2 Function: getDate

Displays the current date in the format mm/dd/yyy Default time zone is UTC.

Author

Farhan Shahbaz

Here is the call graph for this function:

4.23.1.3 parseDate()

```
void parseDate (
            char * date,
            int * m,
            int * d,
            int * y )
```

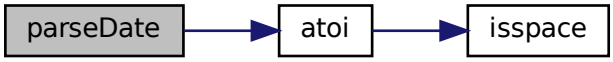
4.23.1.4 Function: parseDate

Parser is used to get and seperate the digit values from the '/' in the date. Makes sure it is in porper form.

Author

Farhan Shahbaz

Here is the call graph for this function:



4.23.1.5 setDate()

```
void setDate (
             int day,
             int month,
             int year )
```

4.23.1.6 Function: setDate

Prompts the user for what date they want to set. The function accesses the day, month, and year registers. Needs to be in the dd/mm/yyyy format.

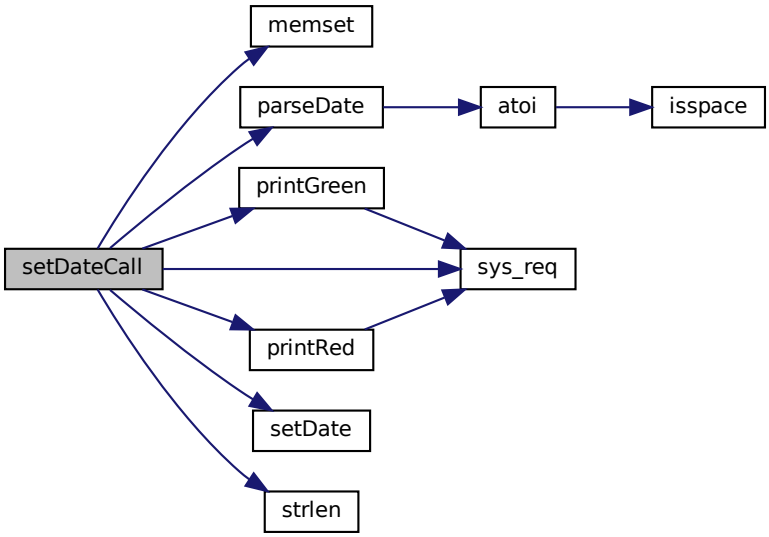
Author

Farhan Shahbaz

4.23.1.7 setDateCall()

```
void setDateCall ( )
```

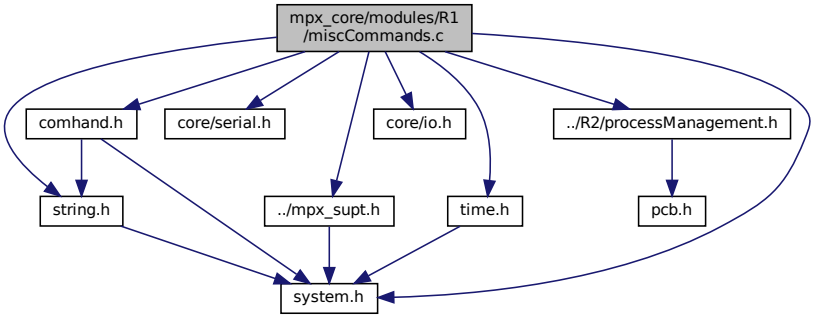
Here is the call graph for this function:



4.24 mpx_core/modules/R1/miscCommands.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include "../mpx_supt.h"
#include <core/io.h>
#include "comhand.h"
#include "../R2/processManagement.h"
#include "time.h"
```

Include dependency graph for miscCommands.c:



Macros

- #define CLEAR_ALL "\033[2J"
- #define MOVE_DEFAULT "\033[0;0H"

Functions

- void getTimeCall ()
- void clear ()
- void version ()
- void help2 ()
- void help3 ()
- void help4 ()
- void help5 ()
- void help ()
- int shutdown ()
- void showPCBCall ()

4.24.1 Macro Definition Documentation

4.24.1.1 CLEAR_ALL

```
#define CLEAR_ALL "\033[2J"
```

4.24.1.2 MOVE_DEFAULT

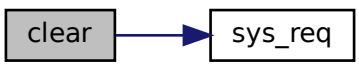
```
#define MOVE_DEFAULT "\033[0;0H"
```

4.24.2 Function Documentation

4.24.2.1 clear()

```
void clear ( )
```

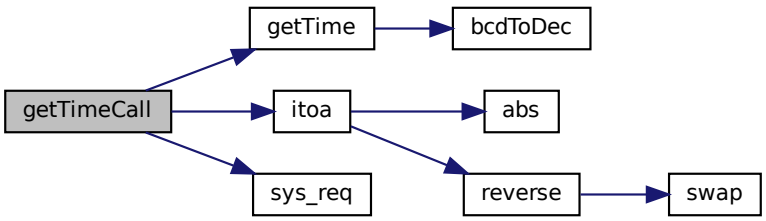
Here is the call graph for this function:



4.24.2.2 getTimeCall()

```
void getTimeCall ( )
```

Here is the call graph for this function:



4.24.2.3 help()

```
void help ( )
```

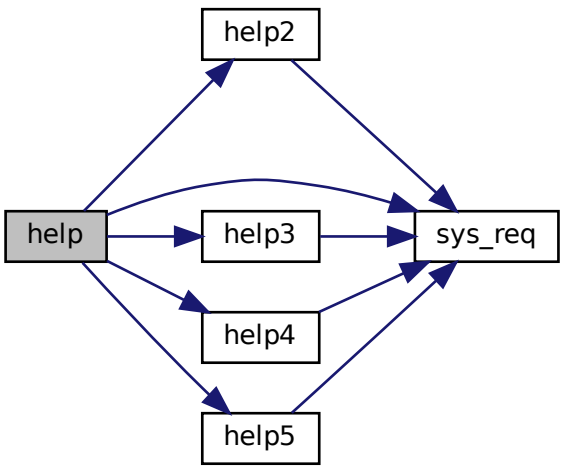
4.24.2.4 Function: help

print statement of the current commands that are executable. Tells user what commands do.

Author

Bryce Williams

Here is the call graph for this function:



4.24.2.5 help2()

```
void help2 ( )
```

Here is the call graph for this function:

4.24.2.6 help3()

```
void help3 ( )
```

Here is the call graph for this function:

4.24.2.7 help4()

```
void help4 ( )
```

Here is the call graph for this function:

4.24.2.8 help5()

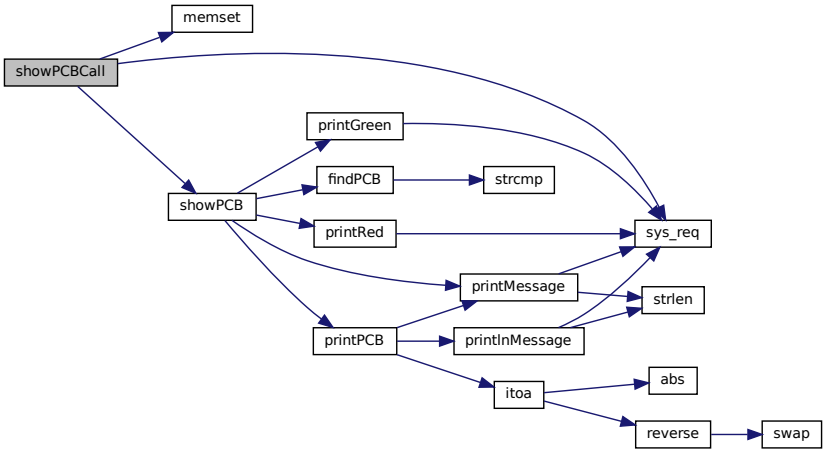
```
void help5 ( )
```

Here is the call graph for this function:

4.24.2.9 showPCBCall()

```
void showPCBCall ( )
```

Here is the call graph for this function:



4.24.2.10 shutdown()

```
int shutdown ( )
```

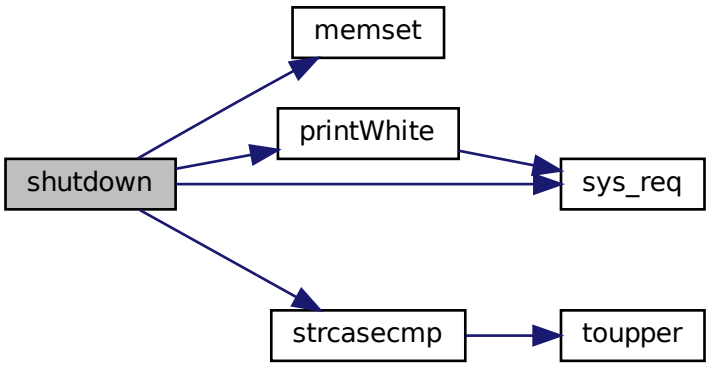
4.24.2.11 Function: shutdown

Shuts downt the comhand by asking the user for a confirmation first

Author

Farhan Shahbaz

Here is the call graph for this function:



4.24.2.12 version()

```
void version ( )
```

4.24.2.13 Function: version

Displays the current version of the project.

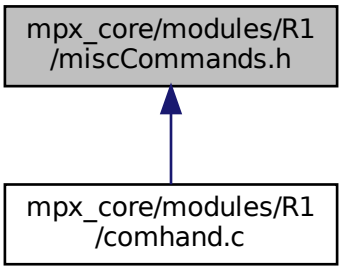
Author

Bryce Williams

Here is the call graph for this function:

4.25 mpx_core/modules/R1/miscCommands.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

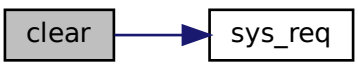
- void version ()
- void help ()
- void help2 ()
- void help3 ()
- void help4 ()
- void help5 ()
- void clear ()
- void getTimeCall ()
- int shutdown ()
- void showPCBCall ()

4.25.1 Function Documentation

4.25.1.1 clear()

```
void clear ( )
```

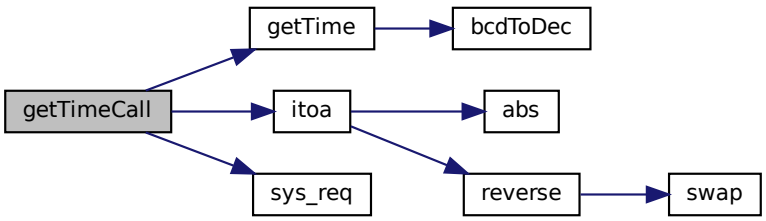
Here is the call graph for this function:



4.25.1.2 getTimeCall()

```
void getTimeCall ( )
```

Here is the call graph for this function:



4.25.1.3 help()

```
void help ( )
```

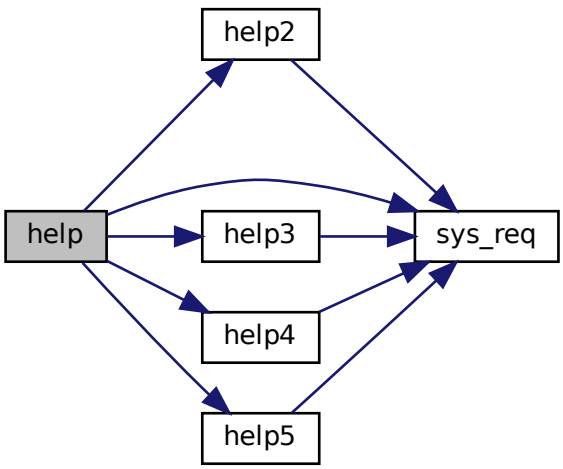
4.25.1.4 Function: help

print statement of the current commands that are executable. Tells user what commands do.

Author

Bryce Williams

Here is the call graph for this function:



4.25.1.5 help2()

```
void help2 ( )
```

Here is the call graph for this function:

4.25.1.6 help3()

```
void help3 ( )
```

Here is the call graph for this function:

4.25.1.7 help4()

```
void help4 ( )
```

Here is the call graph for this function:

4.25.1.8 help5()

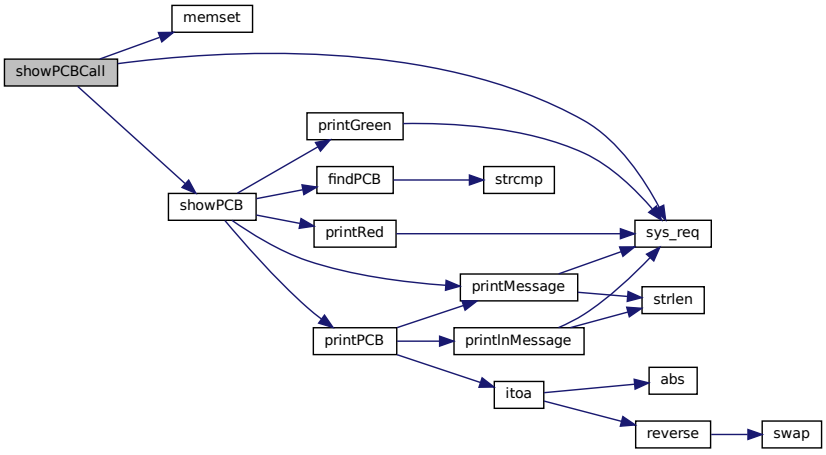
```
void help5 ( )
```

Here is the call graph for this function:

4.25.1.9 showPCBCall()

```
void showPCBCall ( )
```

Here is the call graph for this function:



4.25.1.10 shutdown()

```
int shutdown ( )
```

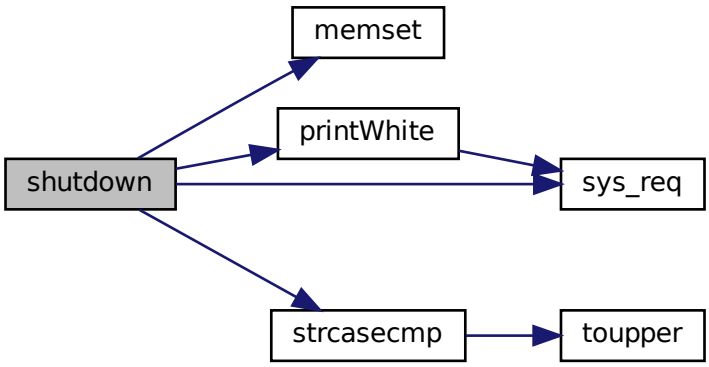
4.25.1.11 Function: shutdown

Shuts downt the comhand by asking the user for a confirmation first

Author

Farhan Shahbaz

Here is the call graph for this function:



4.25.1.12 version()

```
void version ( )
```

4.25.1.13 Function: version

Displays the current version of the project.

Author

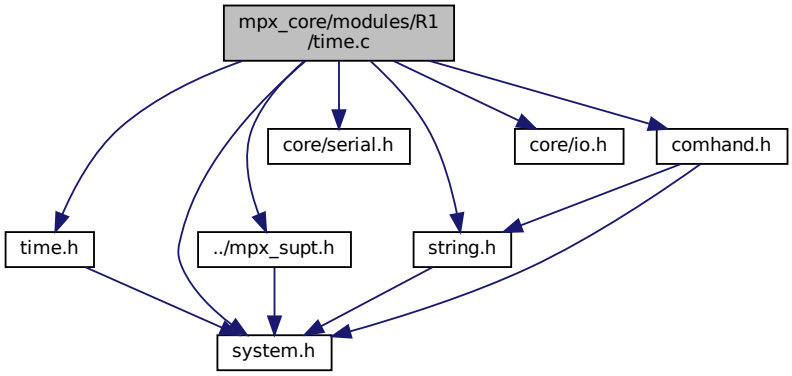
Bryce Williams

Here is the call graph for this function:

4.26 mpx_core/modules/R1/time.c File Reference

```
#include "time.h"
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include "../mpx_supt.h"
#include <core/io.h>
#include "comhand.h"
```

Include dependency graph for time.c:



Functions

- date_time getTime ()
- void setTime (int hours_int, int min_int, int sec_int)
- void parseTime (char *time, int *hh, int *mm, int *ss)
- void setTimeCall ()
- int bcdToDec (int bcd)

4.26.1 Function Documentation

4.26.1.1 bcdToDec()

```
int bcdToDec (
            int bcd )
```

4.26.1.2 getTime()

```
date_time getTime ( )
```

4.26.1.3 Function: getTime

Displays the current time in millitary format: hh:mm:ss. Default time zone is UTC.

Author

Bryce Williams

Here is the call graph for this function:

4.26.1.4 parseTime()

```
void parseTime (
            char * time,
            int * hh,
            int * mm,
            int * ss )
```

4.26.1.5 Function: parseTime

Parses user entered time by replacing ':' array location to terminate null string. atoi() parses time until reaching null character.

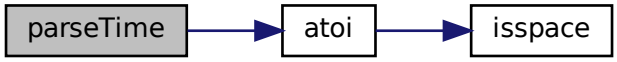
Parameters

time	the time as a character array in the format hh:mm:ss
hh	hours parsed from time are stored in this variable
mm	minutes parsed from time are stored in this variable
ss	seconds parsed from time are stored in this variable

Author

Bryce Williams

Here is the call graph for this function:



4.26.1.6 setTime()

```
void setTime (
            int hours_int,
            int min_int,
            int sec_int )
```

4.26.1.7 Function: setTime

Prompts the user to enter time in the format hh:mm:ss Changes the time in the intel clock register to user set time for displaying with getTime()

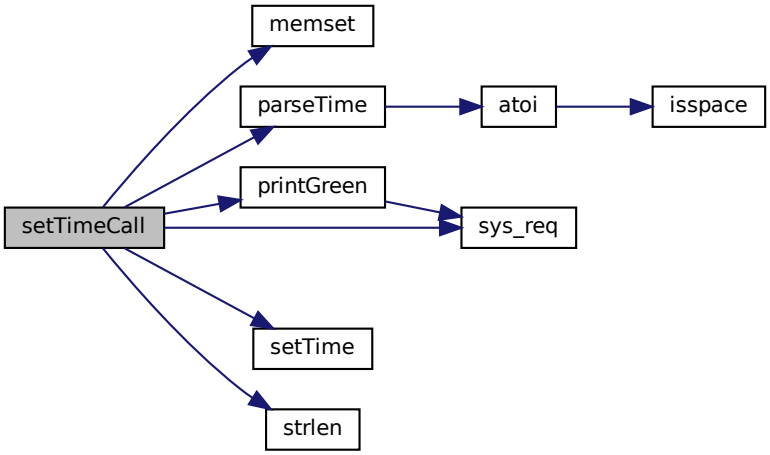
Author

Bryce Williams

4.26.1.8 setTimeCall()

```
void setTimeCall ( )
```

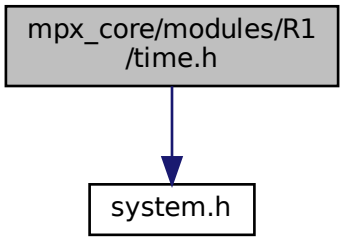
Here is the call graph for this function:



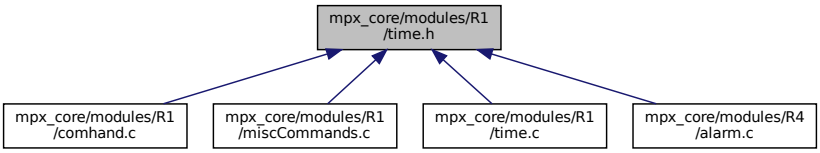
4.27 mpx_core/modules/R1/time.h File Reference

```
#include <system.h>
```

Include dependency graph for time.h:



This graph shows which files directly or indirectly include this file:



Functions

- date_time getTime ()
- void setTime (int hours_int, int min_int, int sec_int)
- void parseTime (char *time, int *hh, int *mm, int *ss)
- void setTimeCall ()
- int bcdToDec (int bcd)

4.27.1 Function Documentation

4.27.1.1 bcdToDec()

```
int bcdToDec (
            int bcd )
```

4.27.1.2 getTime()

```
date_time getTime ( )
```

4.27.1.3 Function: getTime

Displays the current time in millitary format: hh:mm:ss. Default time zone is UTC.

Author

Bryce Williams

Here is the call graph for this function:

4.27.1.4 parseTime()

```
void parseTime (
            char * time,
            int * hh,
            int * mm,
            int * ss )
```

4.27.1.5 Function: parseTime

Parses user entered time by replacing ':' array location to terminate null string. atoi() parses time until reaching null character.

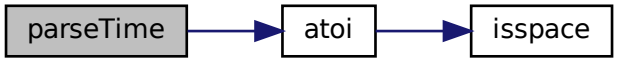
Parameters

time	the time as a character array in the format hh:mm:ss
hh	hours parsed from time are stored in this variable
mm	minutes parsed from time are stored in this variable
ss	seconds parsed from time are stored in this variable

Author

Bryce Williams

Here is the call graph for this function:



4.27.1.6 setTime()

```
void setTime (
             int hours_int,
             int min_int,
             int sec_int )
```

4.27.1.7 Function: setTime

Prompts the user to enter time in the format hh:mm:ss Changes the time in the intel clock register to user set time for displaying with getTime()

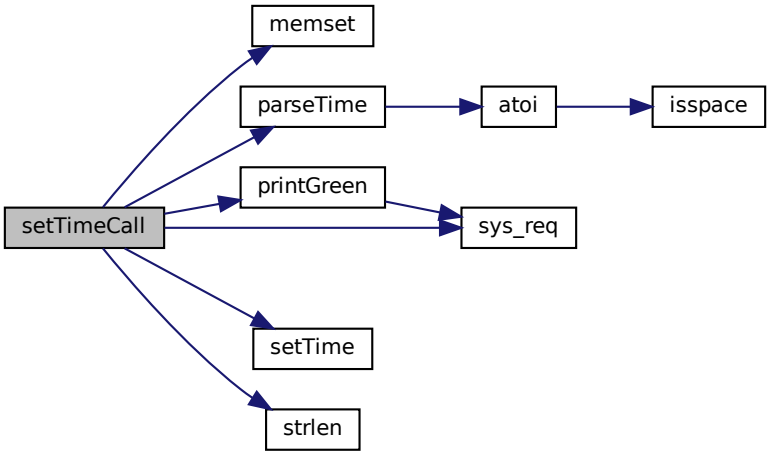
Author

Bryce Williams

4.27.1.8 setTimeCall()

```
void setTimeCall ( )
```

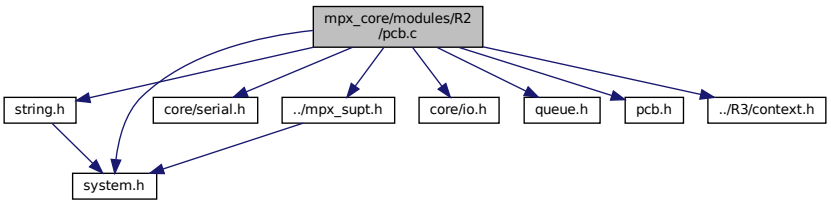
Here is the call graph for this function:



4.28 mpx_core/modules/R2/pcb.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include "../mpx_supt.h"
#include <core/io.h>
#include "queue.h"
#include "pcb.h"
#include "../R3/context.h"
```

Include dependency graph for pcb.c:



Functions

- PCB ∗ allocatePCB ()
- int freePCB (PCB ∗p)
- PCB ∗ setupPCB (char ∗name, int class, int priority)
- void printPCB (PCB ∗pcb_p)

4.28.1 Function Documentation

4.28.1.1 allocatePCB()

```
PCB* allocatePCB ( )
```

4.28.1.2 Function: allocatePCB

Dynamically allocates a PCB.

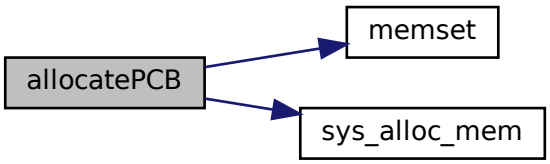
Returns

the allocated PCB

Author

Brendan Michael

Here is the call graph for this function:



4.28.1.3 freePCB()

```
int freePCB (
            PCB * p )
```

4.28.1.4 Function: freePCB

Frees memory associated with a PCB

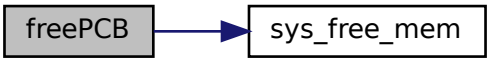
Returns

1 if no error

Author

Brendan Michael

Here is the call graph for this function:



4.28.1.5 printPCB()

```
void printPCB (
            PCB * pcb_p )
```

4.28.1.6 Function: printPCB

Displays the name, status, and class of the selected PCB.

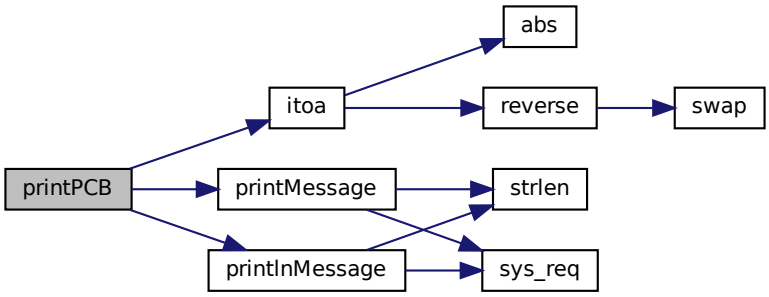
Parameters

pcb_p	pointer of a declared PCB.

Author

Bryce Williams

Here is the call graph for this function:



4.28.1.7 setupPCB()

```
PCB* setupPCB (
            char * name,
            int class,
            int priority )
```

4.28.1.8 Function: setupPCB

Sets a PCBS fields according to the provided parameters

Parameters

∗name	the name for the PCB
class	int class value to be assigned
priority	int priority value to be assigned

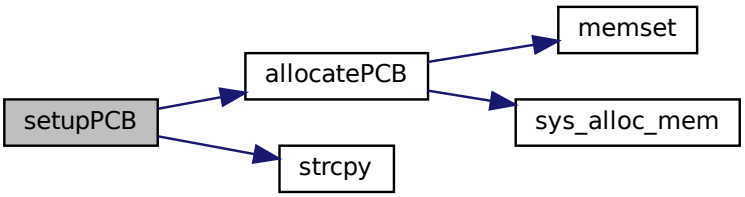
Returns

a pointer to the set up PCB

Author

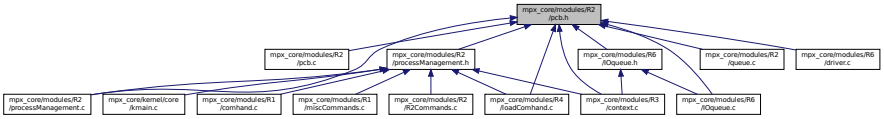
Brendan Michael

Here is the call graph for this function:



4.29 mpx_core/modules/R2/pcb.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct PCB

Macros

- #define READY 0
- #define RUNNING 1
- #define BLOCKED 2
- #define SUSPENDED 1
- #define NOTSUSPENDED 0
- #define SYSTEM 0
- #define APPLICATION 1
- #define stackSize 1024

Typedefs

- typedef struct PCB PCB

Functions

- PCB ∗ allocatePCB ()
- int freePCB (PCB ∗p)
- PCB ∗ setupPCB (char ∗name, int class, int priority)
- void printPCB (PCB ∗pcb_p)

4.29.1 Macro Definition Documentation

4.29.1.1 APPLICATION

```
#define APPLICATION 1
```

4.29.1.2 BLOCKED

```
#define BLOCKED 2
```

4.29.1.3 NOTSUSPENDED

```
#define NOTSUSPENDED 0
```

4.29.1.4 READY

```
#define READY 0
```

4.29.1.5 RUNNING

```
#define RUNNING 1
```

4.29.1.6 stackSize

```
#define stackSize 1024
```

4.29.1.7 SUSPENDED

```
#define SUSPENDED 1
```

4.29.1.8 SYSTEM

```
#define SYSTEM 0
```

4.29.2 Typedef Documentation

4.29.2.1 PCB

```
typedef struct PCB PCB
```

4.29.3 Function Documentation

4.29.3.1 allocatePCB()

```
PCB* allocatePCB ( )
```

4.29.3.2 Function: allocatePCB

Dynamically allocates a PCB.

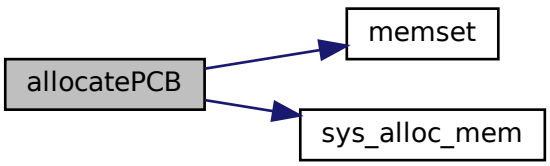
Returns

the allocated PCB

Author

Brendan Michael

Here is the call graph for this function:



4.29.3.3 freePCB()

```
int freePCB (
            PCB * p )
```

4.29.3.4 Function: freePCB

Frees memory associated with a PCB

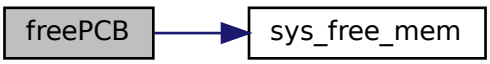
Returns

1 if no error

Author

Brendan Michael

Here is the call graph for this function:



4.29.3.5 printPCB()

```
void printPCB (
            PCB * pcb_p )
```

4.29.3.6 Function: printPCB

Displays the name, status, and class of the selected PCB.

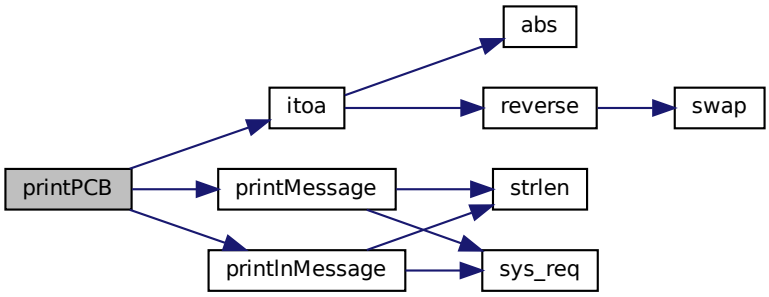
Parameters

pcb_p	pointer of a declared PCB.

Author

Bryce Williams

Here is the call graph for this function:



4.29.3.7 setupPCB()

```
PCB* setupPCB (
            char * name,
            int class,
            int priority )
```

4.29.3.8 Function: setupPCB

Sets a PCBS fields according to the provided parameters

Parameters

∗name	the name for the PCB
class	int class value to be assigned
priority	int priority value to be assigned

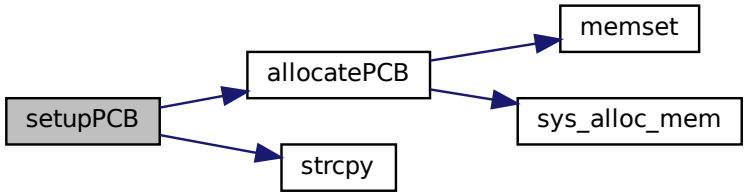
Returns

a pointer to the set up PCB

Author

Brendan Michael

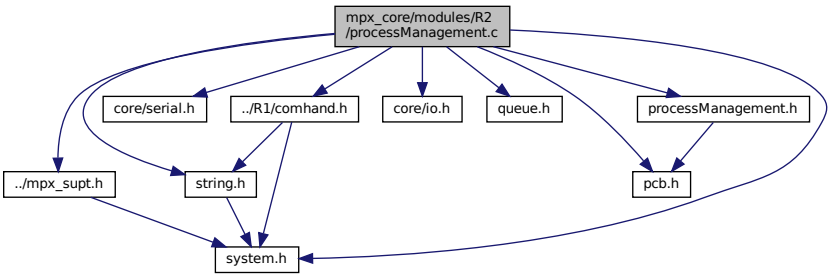
Here is the call graph for this function:



4.30 mpx_core/modules/R2/processManagement.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include "../mpx_supt.h"
#include <core/io.h>
#include "queue.h"
#include "pcb.h"
#include "processManagement.h"
#include "../R1/comhand.h"
```

Include dependency graph for processManagement.c:



Functions

- Queue * createQueue (char *name)
- PCB * getNextReady ()
- void noQueueError ()
- int clearQueues ()
- void initQueues ()
- int deletePCB (char *name)
- void insertPCB (PCB *pcb)
- void createPCB (char *name, int class, int priority)
- int showReady ()

- void setPCBpriority (char *name, int priority)
- void showAll ()
- void showBlocked ()
- void removeReady (PCB *p)
- void removeBlocked (PCB *p)
- int blockPCB (char *name)
- int unblockPCB (char *name)
- int resumePCB (char *name)
- int suspendPCB (char *name)
- void showPCB (char *name)

Variables

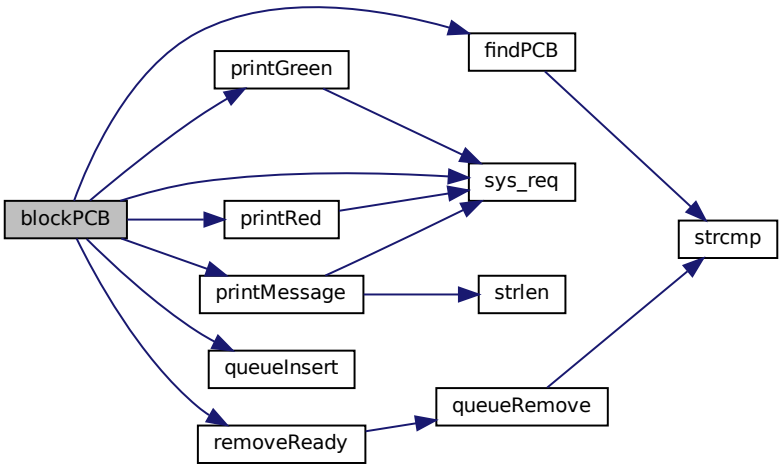
- Queue * readyQueue
- Queue * blockedQueue
- int queuesCleared

4.30.1 Function Documentation

4.30.1.1 blockPCB()

```
int blockPCB (
            char * name )
```

Here is the call graph for this function:



4.30.1.2 clearQueues()

```
int clearQueues ( )
```

4.30.1.3 createPCB()

```
void createPCB (
            char * name,
            int class,
            int priority )
```

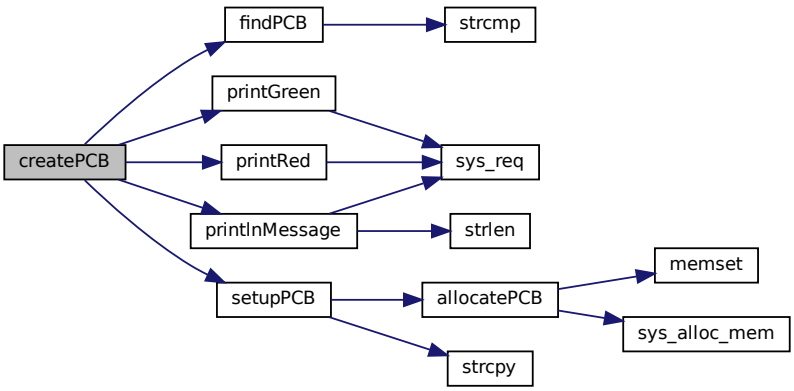
4.30.1.4 Function: createPCB

Allocates memory for a new PCB

Author

Brendan Michael

Here is the call graph for this function:



4.30.1.5 createQueue()

```
Queue* createQueue (
            char * name )
```

4.30.1.6 Function: showBlocked

Displays the blocked queue

Returns

1 if no error

Author

Brendan Michael Function: createPCB

Creates PCB with user specifies name and priority. If user enters a name already in use, display error message.

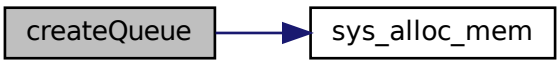
Parameters

name	user entered name of PCB
class	int value representing class of PCB
priority	int value representing priority of PCB

Author

Bryce Williams

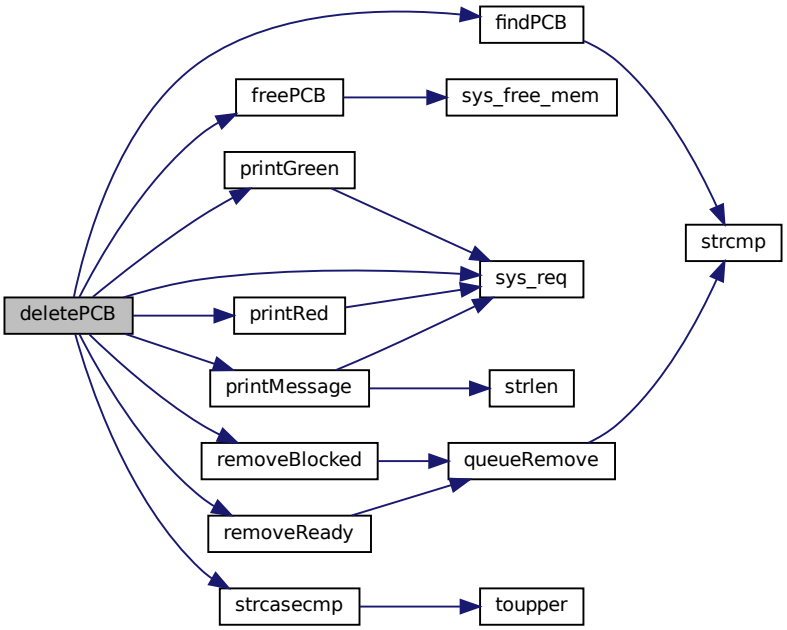
Here is the call graph for this function:



4.30.1.7 deletePCB()

```
int deletePCB (
            char * name )
```

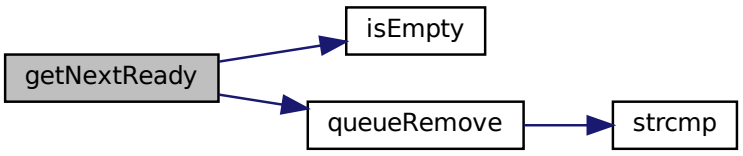
Here is the call graph for this function:



4.30.1.8 getNextReady()

```
PCB* getNextReady ( )
```

Here is the call graph for this function:



4.30.1.9 initQueues()

```
void initQueues ( )
```

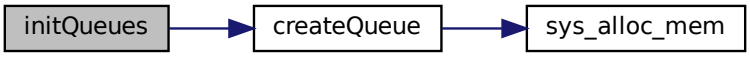
4.30.1.10 Function: initQueues

Initializes the ready and blocked queues

Author

Brendan Michael

Here is the call graph for this function:



4.30.1.11 insertPCB()

```
void insertPCB (
            PCB * pcb )
```

Function: showBlocked

Displays queues that are currently in the blocked state.
If nothing is in block queue displays message that
the queue is currently empty.

Author

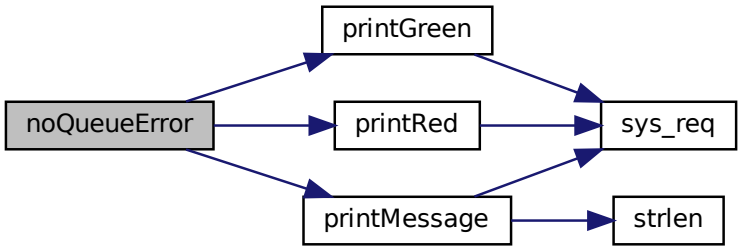
Bryce Williams

Here is the call graph for this function:

4.30.1.12 noQueueError()

```
void noQueueError ( )
```

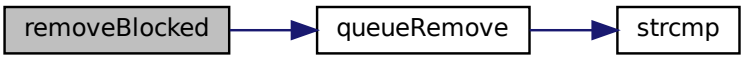
Here is the call graph for this function:



4.30.1.13 removeBlocked()

```
void removeBlocked (
            PCB * p )
```

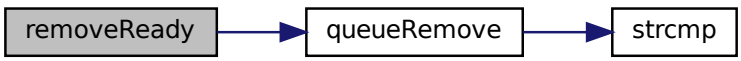
Here is the call graph for this function:



4.30.1.14 removeReady()

```
void removeReady (
            PCB * p )
```

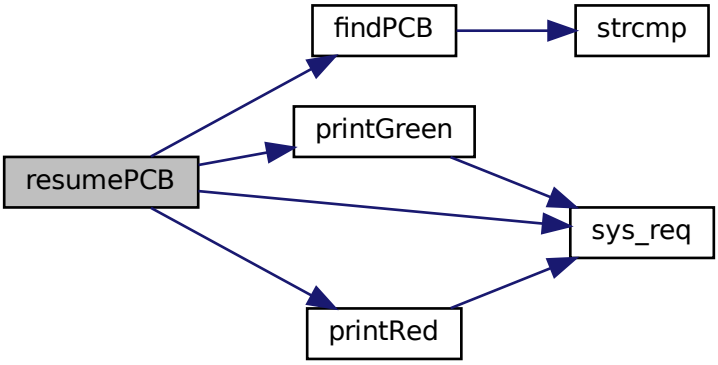
Here is the call graph for this function:



4.30.1.15 resumePCB()

```
int resumePCB (
            char * name )
```

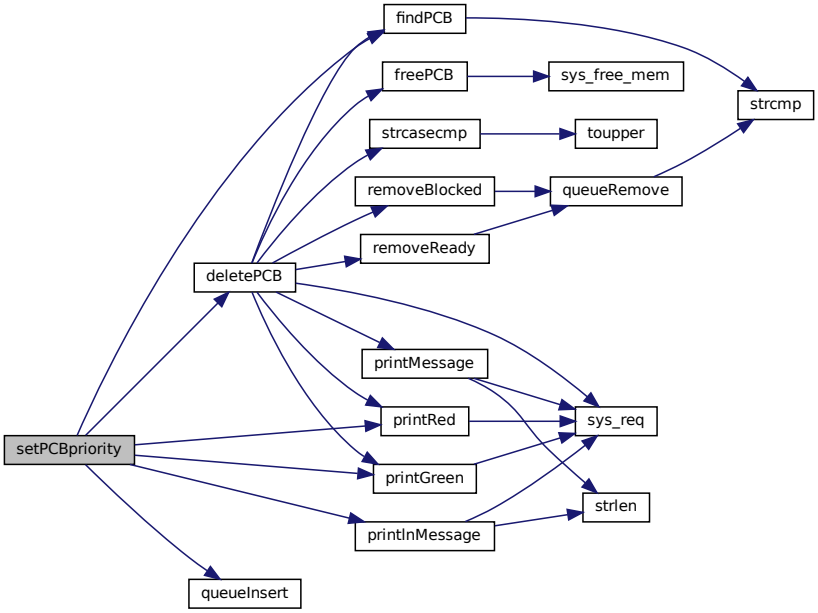
Here is the call graph for this function:



4.30.1.16 setPCBpriority()

```
void setPCBpriority (
            char * name,
            int priority )
```

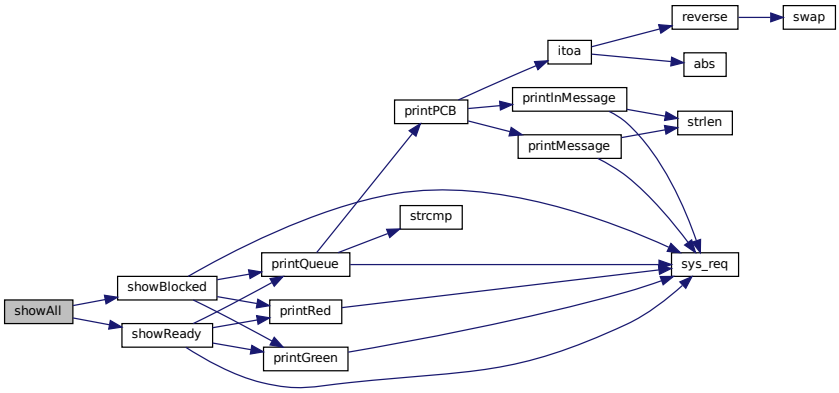
Here is the call graph for this function:



4.30.1.17 showAll()

```
void showAll ( )
```

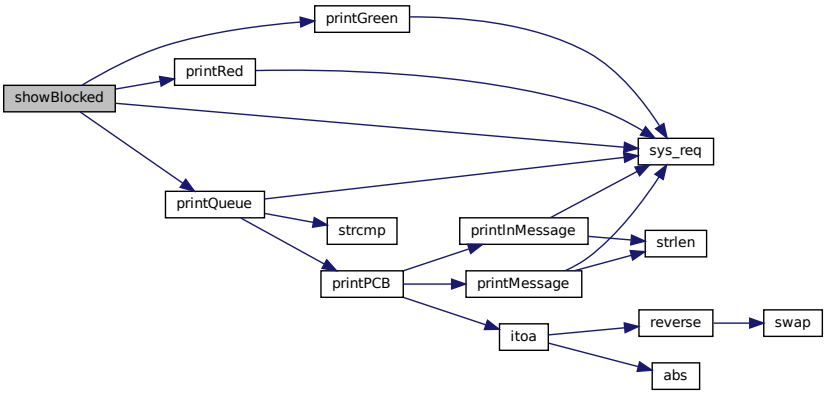
Here is the call graph for this function:



4.30.1.18 showBlocked()

```
void showBlocked ( )
```

Here is the call graph for this function:



4.30.1.19 showPCB()

```
void showPCB (
            char * name )
```

4.30.1.20 Function: showPCB

Finds pcb based on user entered name. Searches in readyQueue then blockedQueue.

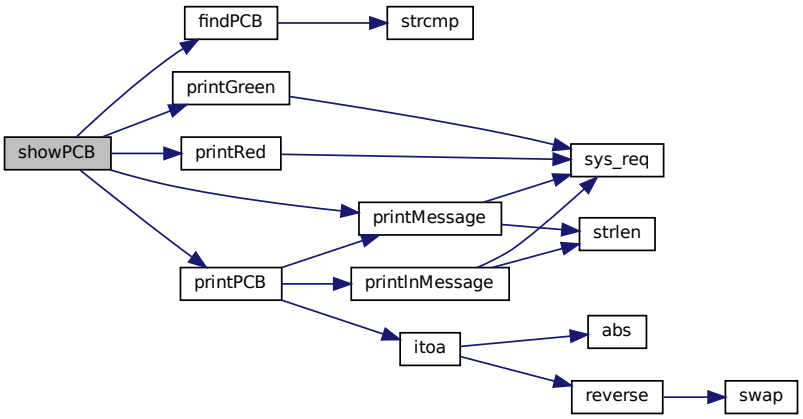
Parameters

name	name of the PCB to search for.

Author

Bryce Williams

Here is the call graph for this function:



4.30.1.21 showReady()

```
int showReady ( )
```

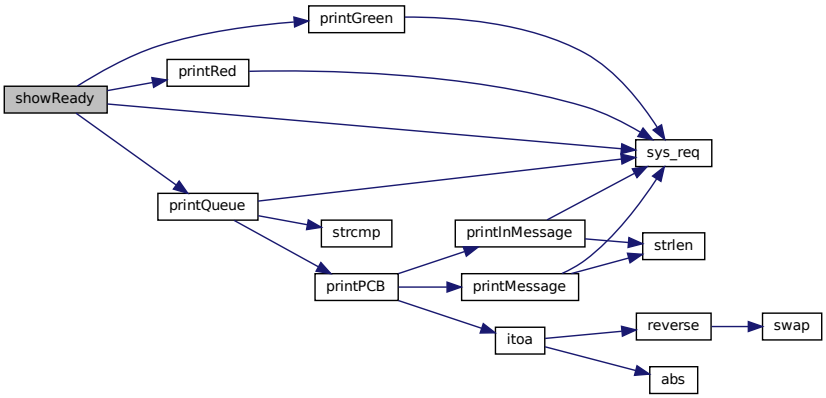
4.30.1.22 Function: showReady

Displays the pcb's currently in the ready state. Checks to see if readyQueue is empty. If it is empty displays an error message.

Author

Bryce Williams

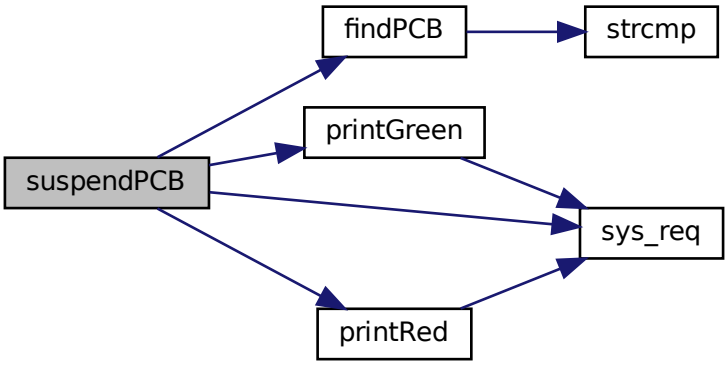
Here is the call graph for this function:



4.30.1.23 suspendPCB()

```
int suspendPCB (
            char * name )
```

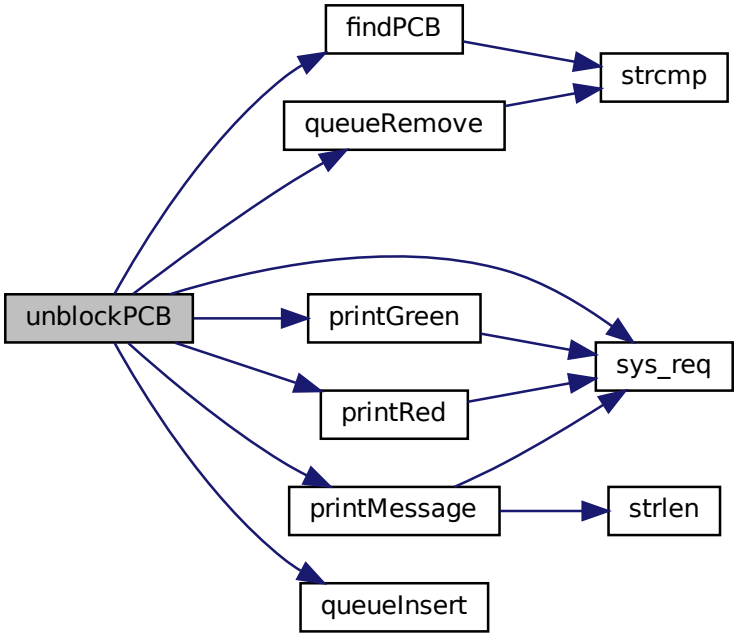
Here is the call graph for this function:



4.30.1.24 unblockPCB()

```
int unblockPCB (
            char * name )
```

Here is the call graph for this function:



4.30.2 Variable Documentation

4.30.2.1 blockedQueue

```
Queue* blockedQueue
```

4.30.2.2 queuesCleared

```
int queuesCleared
```

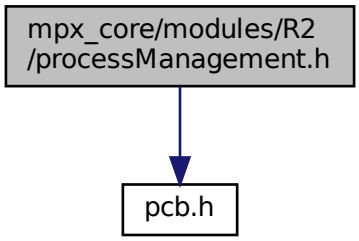
4.30.2.3 readyQueue

```
Queue* readyQueue
```

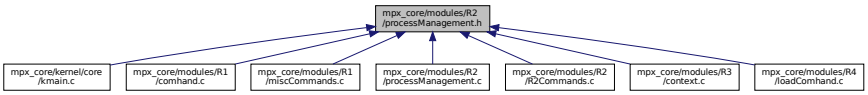
4.31 mpx_core/modules/R2/processManagement.h File Reference

```
#include "pcb.h"
```

Include dependency graph for processManagement.h:



This graph shows which files directly or indirectly include this file:



Functions

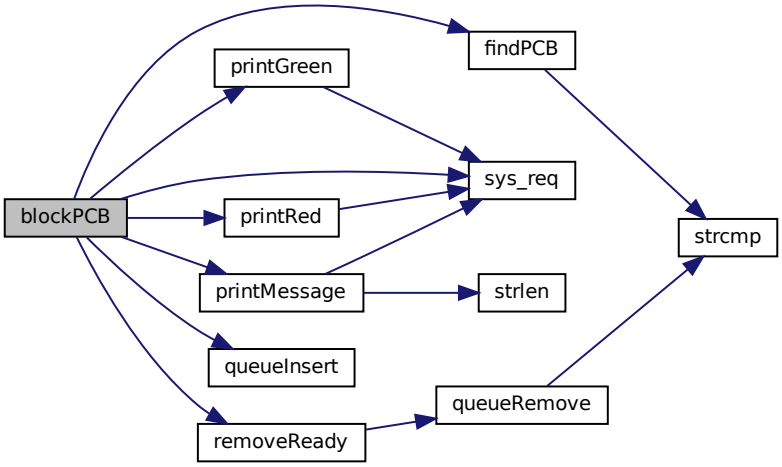
- void initQueues ()
- void createPCB (char ∗name, int class, int priority)
- int showReady ()
- struct Queue ∗ createQueue (char ∗name)
- void showAll ()
- int deletePCB (char ∗name)
- void insertPCB (PCB ∗pcb)
- PCB ∗ getNextReady ()
- void showBlocked ()
- int blockPCB (char ∗name)
- int unblockPCB (char ∗name)
- void showPCB (char ∗name)
- void setPCBpriority (char ∗name, int priority)
- void noQueueError ()
- void removeReady (PCB ∗p)
- void removeBlocked (PCB ∗p)
- int resumePCB (char ∗name)
- int suspendPCB (char ∗name)
- int clearQueues ()

4.31.1 Function Documentation

4.31.1.1 blockPCB()

```
int blockPCB (
            char * name )
```

Here is the call graph for this function:



4.31.1.2 clearQueues()

```
int clearQueues ( )
```

4.31.1.3 createPCB()

```
void createPCB (
            char * name,
            int class,
            int priority )
```

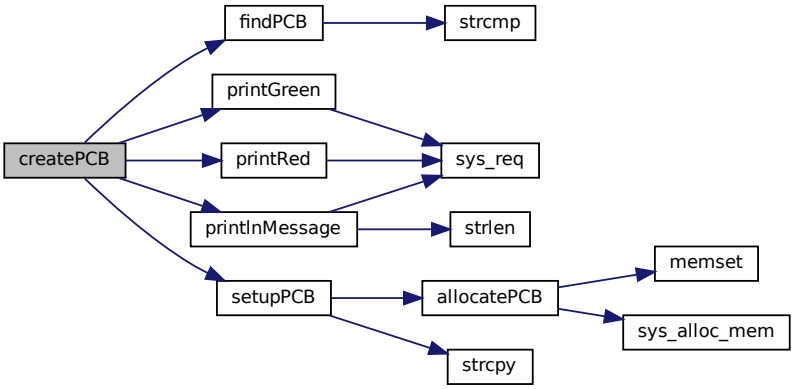
4.31.1.4 Function: createPCB

Allocates memory for a new PCB

Author

Brendan Michael

Here is the call graph for this function:



4.31.1.5 createQueue()

```
struct Queue* createQueue (
            char * name )
```

4.31.1.6 Function: showBlocked

Displays the blocked queue

Returns

1 if no error

Author

Brendan Michael Function: createPCB

Creates PCB with user specifies name and priority. If user enters a name already in use, display error message.

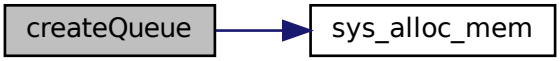
Parameters

name	user entered name of PCB
class	int value representing class of PCB
priority	int value representing priority of PCB

Author

Bryce Williams

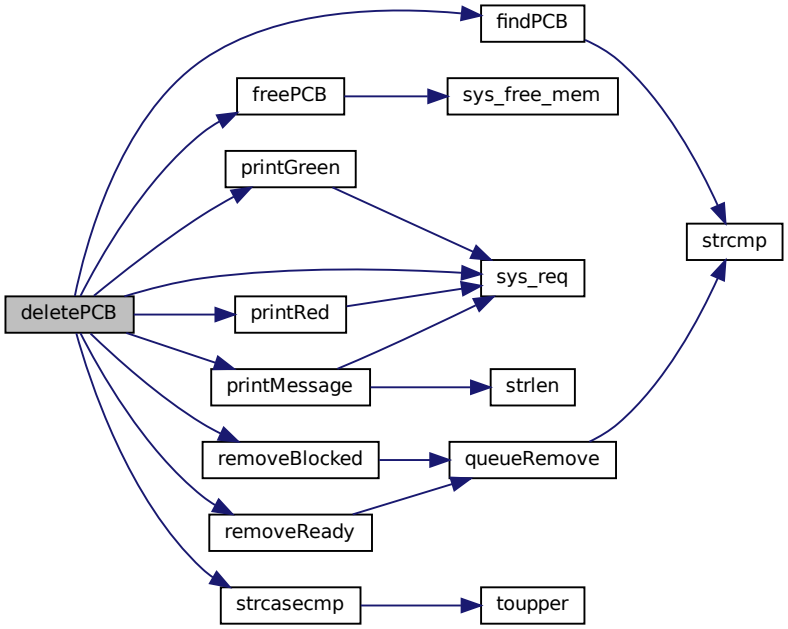
Here is the call graph for this function:



4.31.1.7 deletePCB()

```
int deletePCB (
            char * name )
```

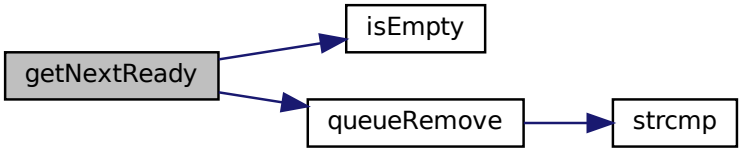
Here is the call graph for this function:



4.31.1.8 getNextReady()

```
PCB* getNextReady ( )
```

Here is the call graph for this function:



4.31.1.9 initQueues()

```
void initQueues ( )
```

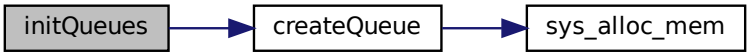
4.31.1.10 Function: initQueues

Initializes the ready and blocked queues

Author

Brendan Michael

Here is the call graph for this function:



4.31.1.11 insertPCB()

```
void insertPCB (
            PCB * pcb )
```

Function: showBlocked

Displays queues that are currently in the blocked state.
If nothing is in block queue displays message that
the queue is currently empty.

Author

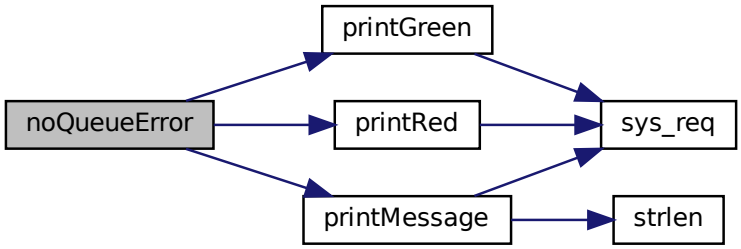
Bryce Williams

Here is the call graph for this function:

4.31.1.12 noQueueError()

```
void noQueueError ( )
```

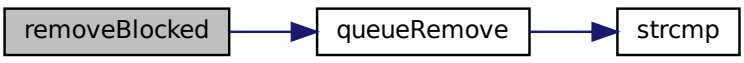
Here is the call graph for this function:



4.31.1.13 removeBlocked()

```
void removeBlocked (
            PCB * p )
```

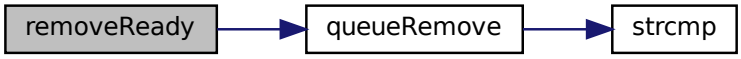
Here is the call graph for this function:



4.31.1.14 removeReady()

```
void removeReady (
            PCB * p )
```

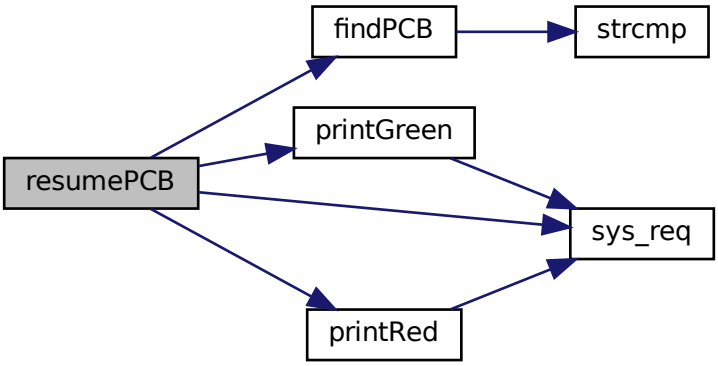
Here is the call graph for this function:



4.31.1.15 resumePCB()

```
int resumePCB (
            char * name )
```

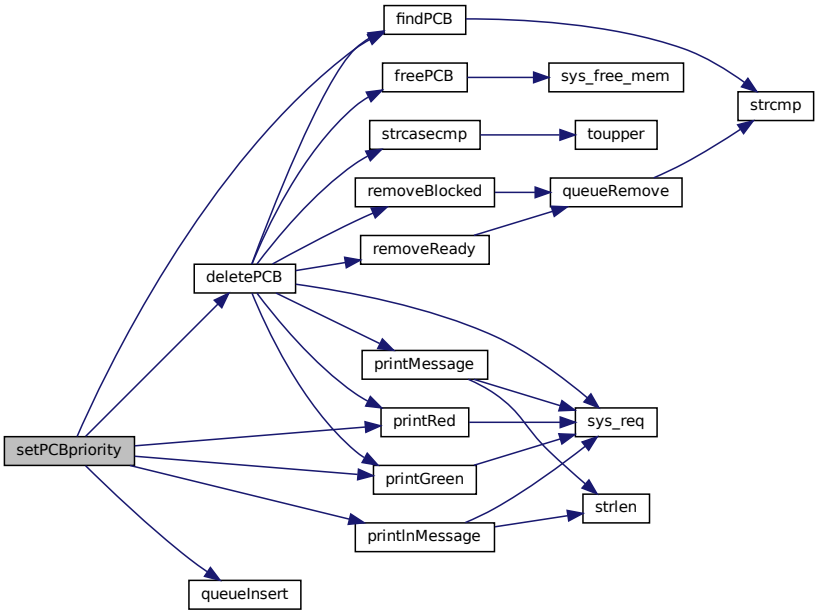
Here is the call graph for this function:



4.31.1.16 setPCBpriority()

```
void setPCBpriority (
            char * name,
            int priority )
```

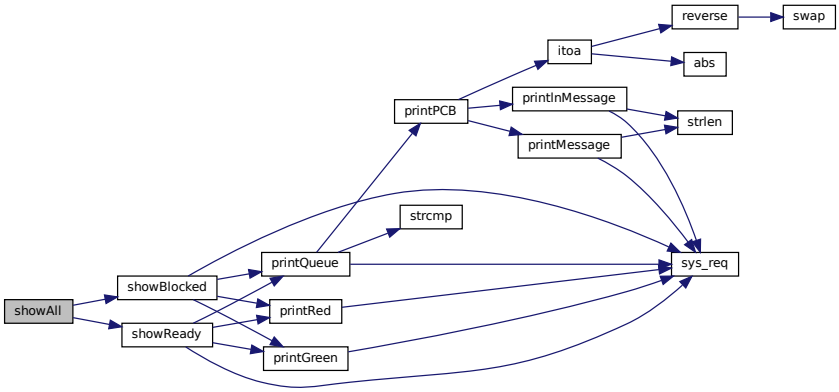
Here is the call graph for this function:



4.31.1.17 showAll()

```
void showAll ( )
```

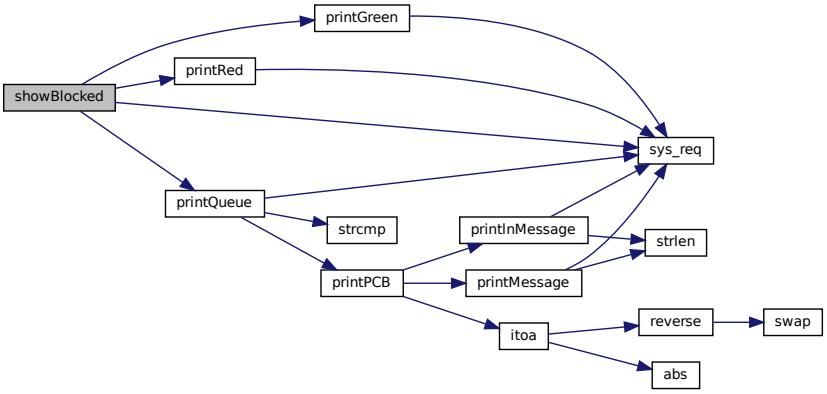
Here is the call graph for this function:



4.31.1.18 showBlocked()

```
void showBlocked ( )
```

Here is the call graph for this function:



4.31.1.19 showPCB()

```
void showPCB (
            char * name )
```

4.31.1.20 Function: showPCB

Finds pcb based on user entered name. Searches in readyQueue then blockedQueue.

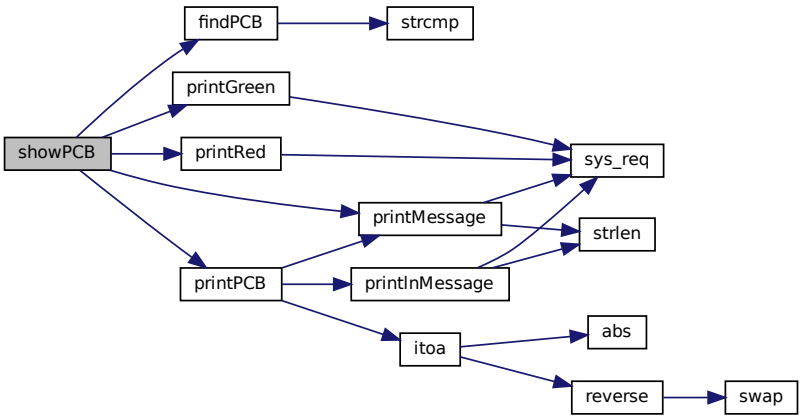
Parameters

name	name of the PCB to search for.

Author

Bryce Williams

Here is the call graph for this function:



4.31.1.21 showReady()

```
int showReady ( )
```

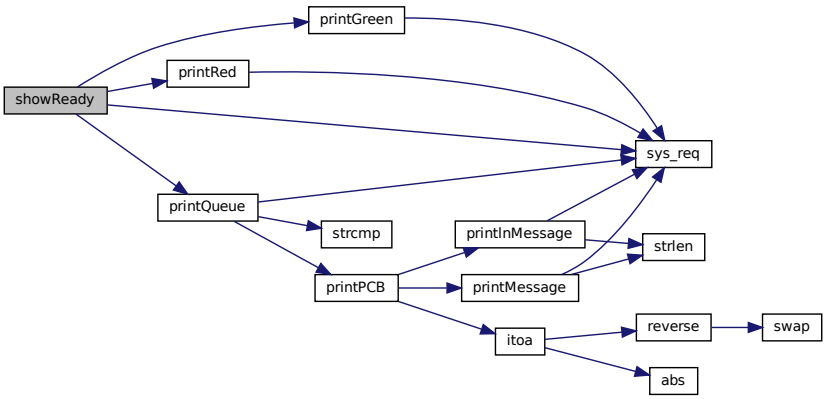
4.31.1.22 Function: showReady

Displays the pcb's currently in the ready state. Checks to see if readyQueue is empty. If it is empty displays an error message.

Author

Bryce Williams

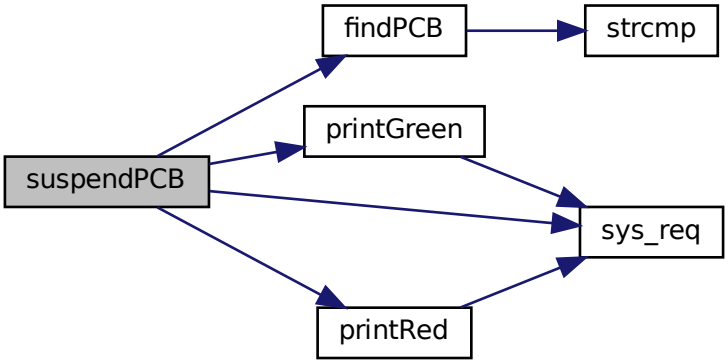
Here is the call graph for this function:



4.31.1.23 suspendPCB()

```
int suspendPCB (
            char * name )
```

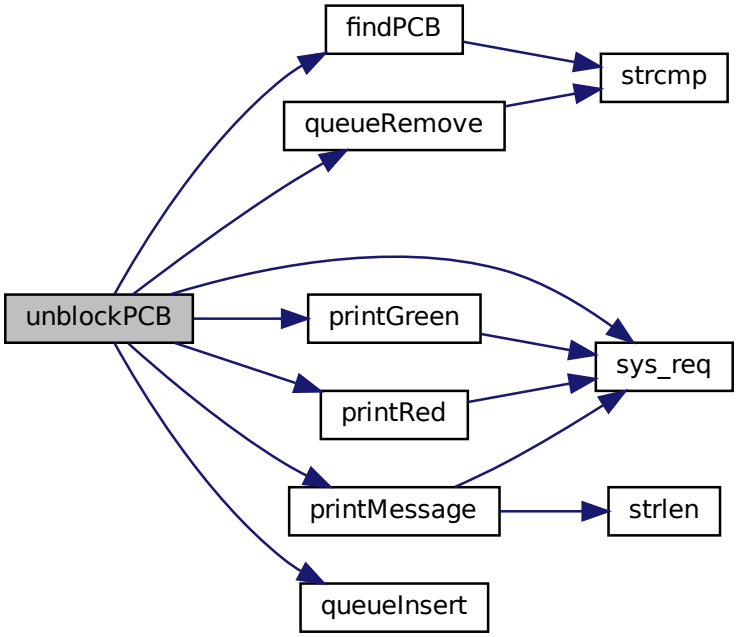
Here is the call graph for this function:



4.31.1.24 unblockPCB()

```
int unblockPCB (
            char * name )
```

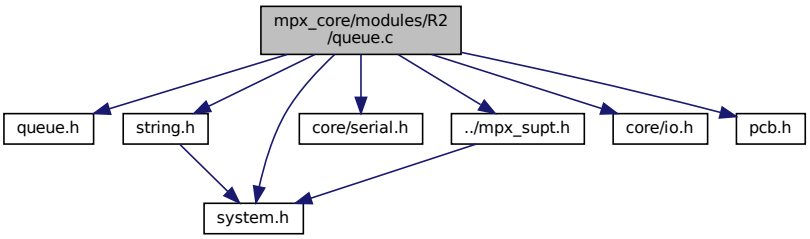
Here is the call graph for this function:



4.32 mpx_core/modules/R2/queue.c File Reference

```
#include "queue.h"
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include "../mpx_supt.h"
#include <core/io.h>
#include "pcb.h"
```

Include dependency graph for queue.c:



Functions

- int isEmpty (Queue *q)
- int queueInsert (Queue *q, PCB *pcb)
- int queueRemove (Queue *q, PCB *pcb)
- void peek (Queue *q)
- PCB * findPCB (char *name, Queue *q)
- void printQueue (Queue *q)

Variables

- char * readyTitle = "READY QUEUE\n"
- char * blockedTitle = "BLOCKED QUEUE\n"
- int titleSize = 20
- char * border = "==========\n"
- int borderSize = 16

4.32.1 Function Documentation

4.32.1.1 findPCB()

```
PCB* findPCB (
            char * name,
            Queue * q )
```

4.32.1.2 Function: findPCB()

Finds and returns a PCB based on the given name and queue.

Parameters

*name	the PCB name
*q	the queue to look through

Returns

the found PCB

Author

Brendan Michael

Here is the call graph for this function:

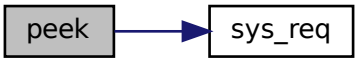
4.32.1.3 isEmpty()

```
int isEmpty (
            Queue * q )
```

4.32.1.4 peek()

```
void peek (
            Queue * q )
```

Here is the call graph for this function:



4.32.1.5 printQueue()

```
void printQueue (
            Queue * q )
```

4.32.1.6 Function: createQueue

Creates and initializes an empty queue.

Returns

the created queue

Author

Brendan Michael

4.32.1.7 Function: findPCB()

Finds and returns a PCB based on the given name and queue.

Parameters

∗name	the PCB name
∗q	the queue to look through

Returns

the found PCB

Author

Brendan Michael

4.32.1.8 Function: printQueue

Prints out the contents of the given queue

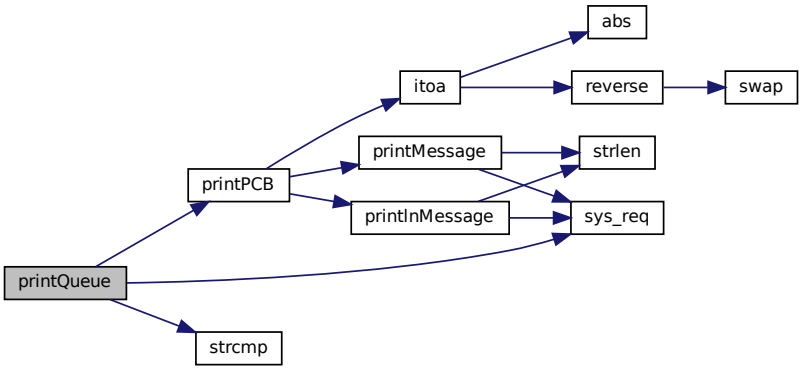
Parameters

∗q	the queue to print out

Author

Brendan Michael

Here is the call graph for this function:



4.32.1.9 queueInsert()

```
int queueInsert (
            Queue * q,
            struct PCB * pcb )
```

4.32.1.10 Function: queueInsert

Parameters

∗q	the queue to be inserted into
∗pcb	the pcb to be inserted

Returns

1 if no error occured

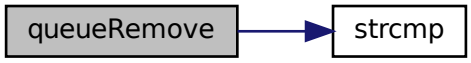
Author

Bryce Williams and Brendan Michael

4.32.1.11 queueRemove()

```
int queueRemove (
            Queue * q,
            PCB * pcb )
```

Here is the call graph for this function:



4.32.2 Variable Documentation

4.32.2.1 blockedTitle

```
char* blockedTitle = "BLOCKED QUEUE\n"
```

4.32.2.2 border

```
char* border = "==============\n"
```

4.32.2.3 borderSize

```
int borderSize = 16
```

4.32.2.4 readyTitle

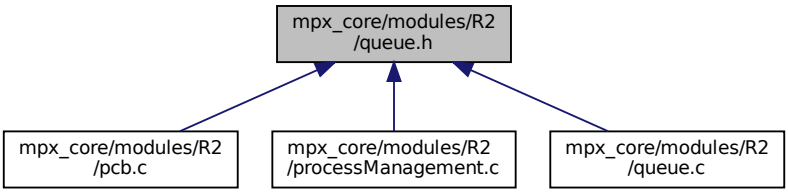
```
char* readyTitle = "READY QUEUE\n"
```

4.32.2.5 titleSize

```
int titleSize = 20
```

4.33 mpx_core/modules/R2/queue.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct Queue

Typedefs

- typedef struct Queue Queue

Functions

- void printQueue (Queue ∗q)
- struct PCB ∗ findPCB (char ∗name, Queue ∗q)
- int queueInsert (Queue ∗q, struct PCB ∗pcb)
- int queueRemove (Queue ∗q, struct PCB ∗pcb)
- int isEmpty (Queue ∗q)
- void peek (Queue ∗q)

4.33.1 Typedef Documentation

4.33.1.1 Queue

```
typedef struct Queue Queue
```

4.33.2 Function Documentation

4.33.2.1 findPCB()

```
struct PCB* findPCB (
            char * name,
            Queue * q )
```

4.33.2.2 Function: findPCB()

Finds and returns a PCB based on the given name and queue.

Parameters

∗*name*	the PCB name
∗*q*	the queue to look through

Returns

the found PCB

Author

Brendan Michael

Here is the call graph for this function:

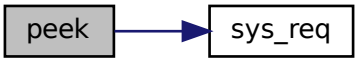
4.33.2.3 isEmpty()

```
int isEmpty (
            Queue * q )
```

4.33.2.4 peek()

```
void peek (
            Queue * q )
```

Here is the call graph for this function:



4.33.2.5 printQueue()

```
void printQueue (
            Queue * q )
```

4.33.2.6 Function: createQueue

Creates and initializes an empty queue.

Returns

the created queue

Author

Brendan Michael

4.33.2.7 Function: findPCB()

Finds and returns a PCB based on the given name and queue.

Parameters

∗name	the PCB name
∗q	the queue to look through

Returns

the found PCB

Author

Brendan Michael

4.33.2.8 Function: printQueue

Prints out the contents of the given queue

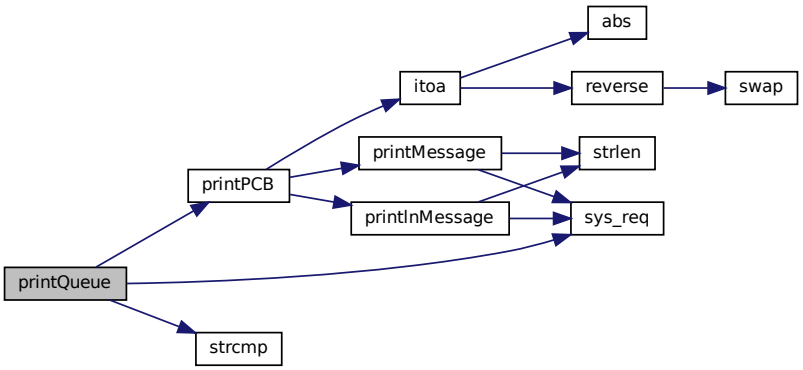
Parameters

∗q	the queue to print out

Author

Brendan Michael

Here is the call graph for this function:



4.33.2.9 queueInsert()

```
int queueInsert (
            Queue * q,
            struct PCB * pcb )
```

4.33.2.10 Function: queueInsert

Parameters

∗q	the queue to be inserted into
∗pcb	the pcb to be inserted

Returns

1 if no error occured

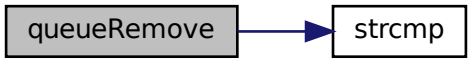
Author

Bryce Williams and Brendan Michael

4.33.2.11 queueRemove()

```
int queueRemove (
            Queue * q,
            struct PCB * pcb )
```

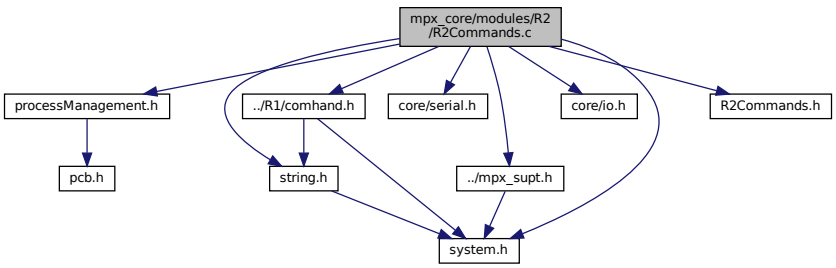
Here is the call graph for this function:



4.34 mpx_core/modules/R2/R2Commands.c File Reference

```
#include "processManagement.h"
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include "../mpx_supt.h"
#include <core/io.h>
#include "../R1/comhand.h"
#include "R2Commands.h"
```

Include dependency graph for R2Commands.c:



Functions

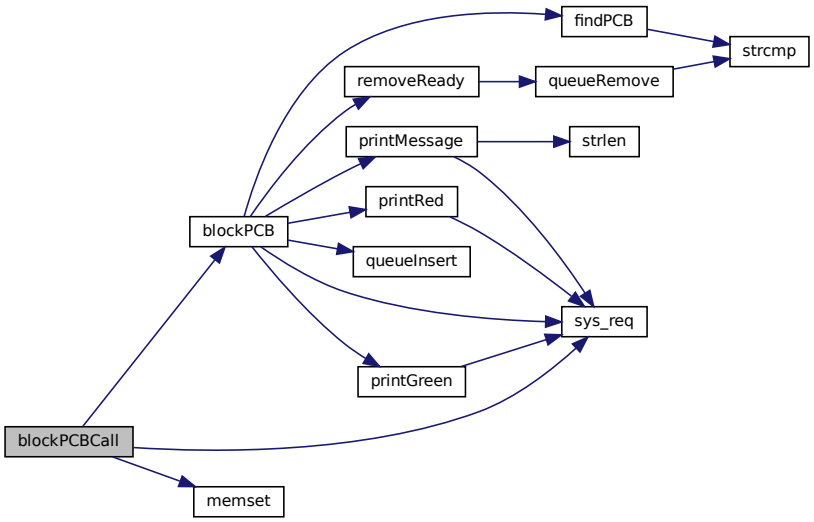
- void showReadyCall ()
- void setPriorityCall ()
- void deletePCBCall ()
- void createPCBCall ()
- void blockPCBCall ()
- void unblockPCBCall ()
- void suspendPCBCall ()
- void resumePCBCall ()

4.34.1 Function Documentation

4.34.1.1 blockPCBCall()

```
void blockPCBCall ( )
```

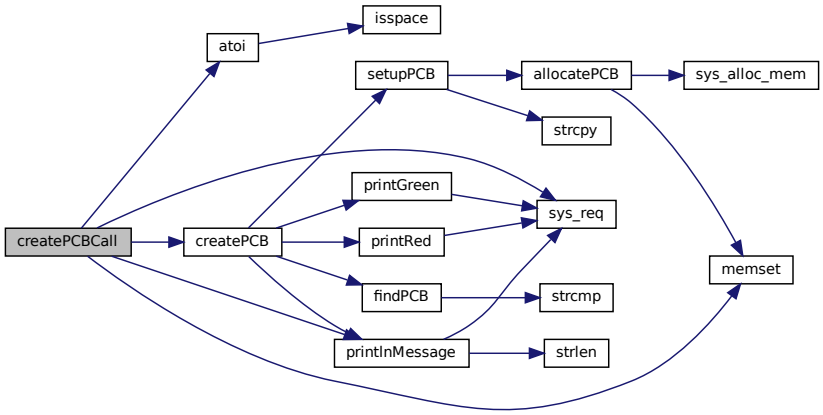
Here is the call graph for this function:



4.34.1.2 createPCBCall()

```
void createPCBCall ( )
```

Here is the call graph for this function:



4.34.1.3 deletePCBCall()

```
void deletePCBCall ( )
```

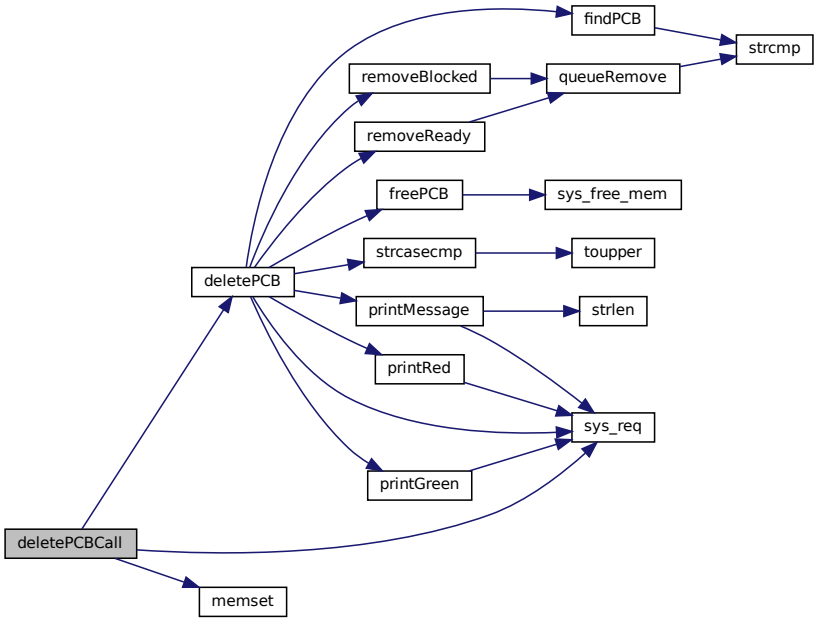
4.34.1.4 Function: noSuchCommand

Displays an error message when an entered command is invalid.

Author

Brendan Michael

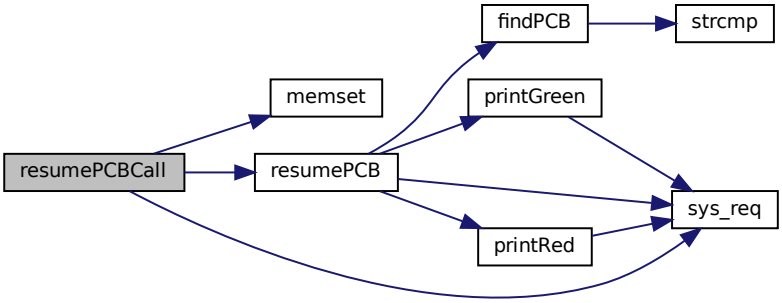
Here is the call graph for this function:



4.34.1.5 resumePCBCall()

```
void resumePCBCall ( )
```

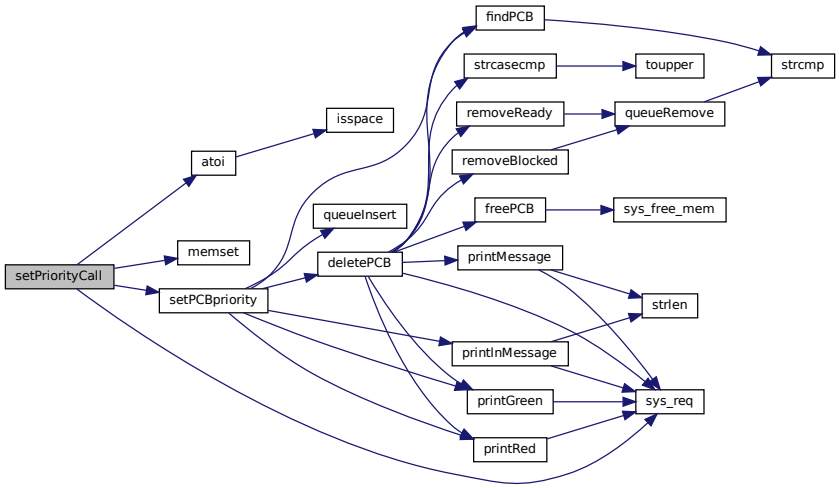
Here is the call graph for this function:



4.34.1.6 setPriorityCall()

```
void setPriorityCall ( )
```

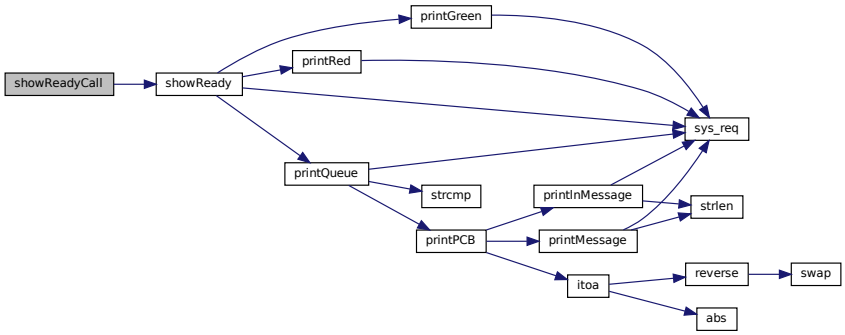
Here is the call graph for this function:



4.34.1.7 showReadyCall()

```
void showReadyCall ( )
```

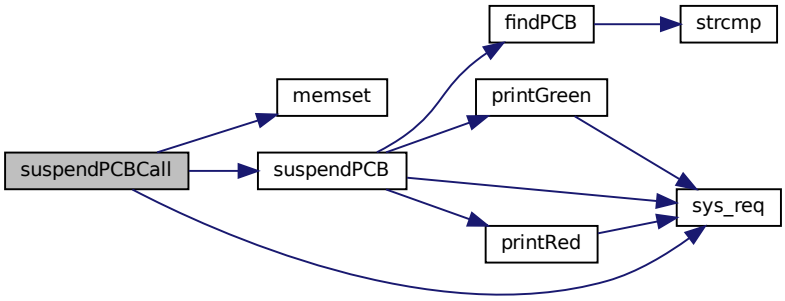
Here is the call graph for this function:



4.34.1.8 suspendPCBCall()

```
void suspendPCBCall ( )
```

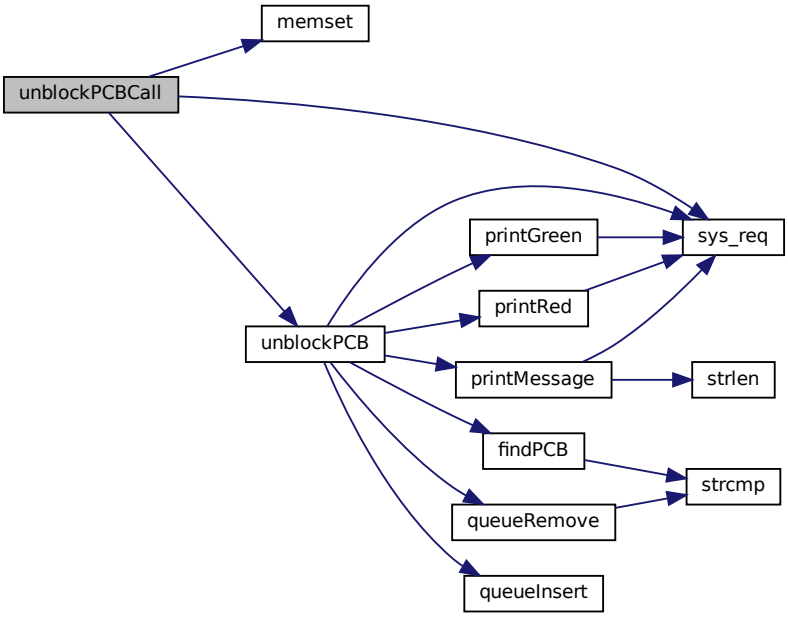
Here is the call graph for this function:



4.34.1.9 unblockPCBCall()

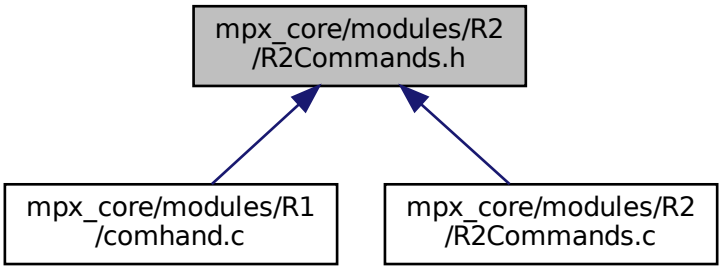
```
void unblockPCBCall ( )
```

Here is the call graph for this function:



4.35 mpx_core/modules/R2/R2Commands.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void deletePCBCall ()
- void createPCBCall ()
- void showReadyCall ()
- void unblockPCBCall ()
- void blockPCBCall ()
- void setPriorityCall ()
- void suspendPCBCall ()

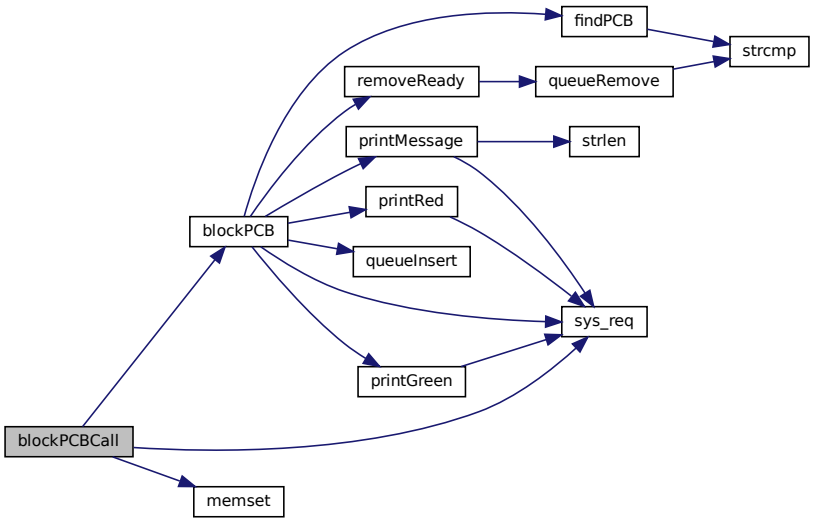
- void resumePCBCall ()

4.35.1 Function Documentation

4.35.1.1 blockPCBCall()

```
void blockPCBCall ( )
```

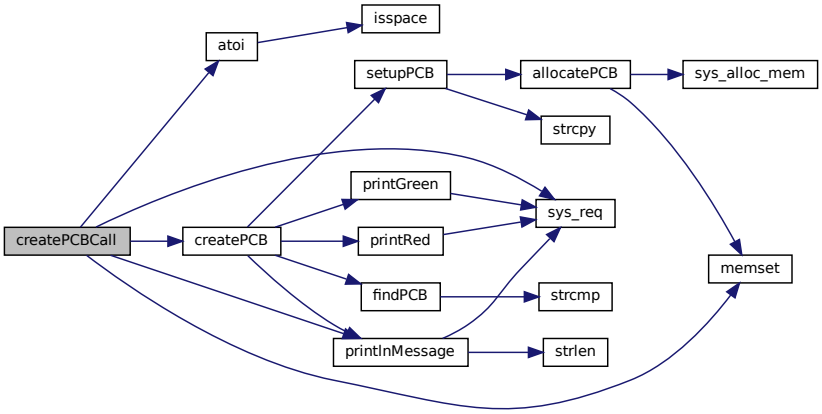
Here is the call graph for this function:



4.35.1.2 createPCBCall()

```
void createPCBCall ( )
```

Here is the call graph for this function:



4.35.1.3 deletePCBCall()

```
void deletePCBCall ( )
```

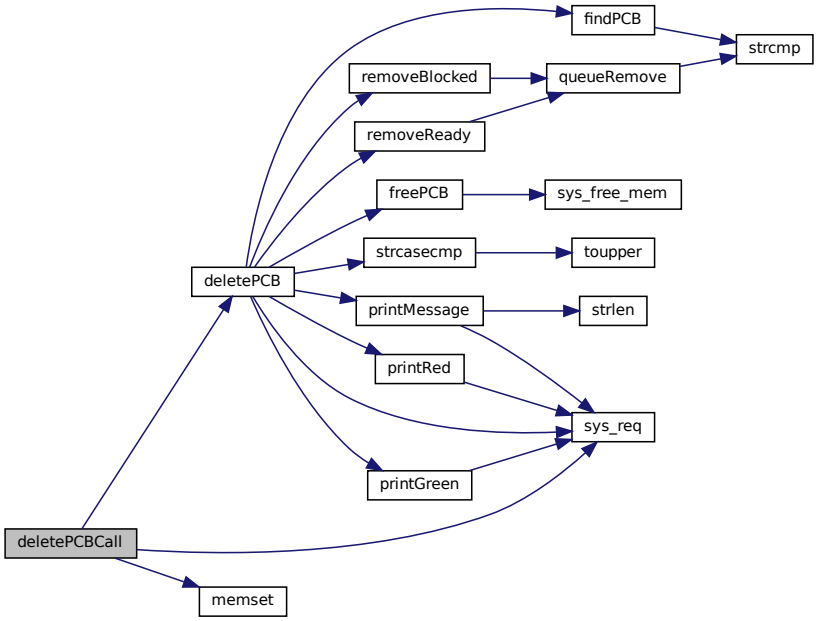
4.35.1.4 Function: noSuchCommand

Displays an error message when an entered command is invalid.

Author

Brendan Michael

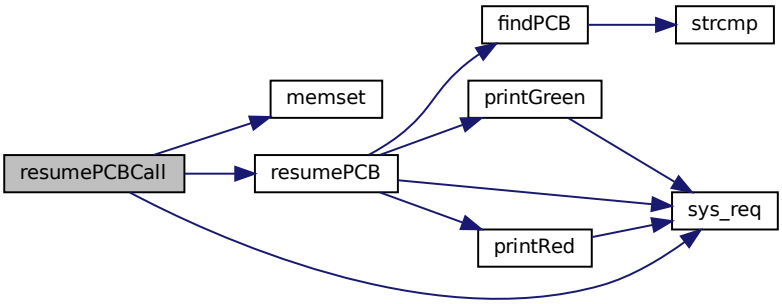
Here is the call graph for this function:



4.35.1.5 resumePCBCall()

```
void resumePCBCall ( )
```

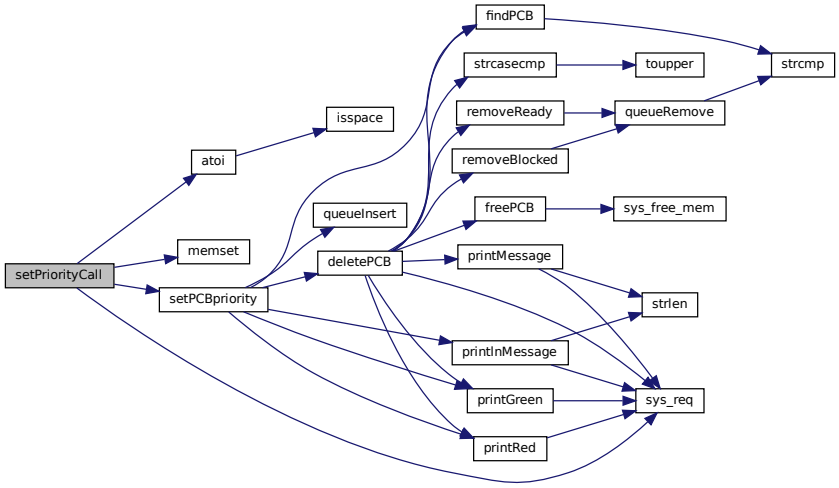
Here is the call graph for this function:



4.35.1.6 setPriorityCall()

```
void setPriorityCall ( )
```

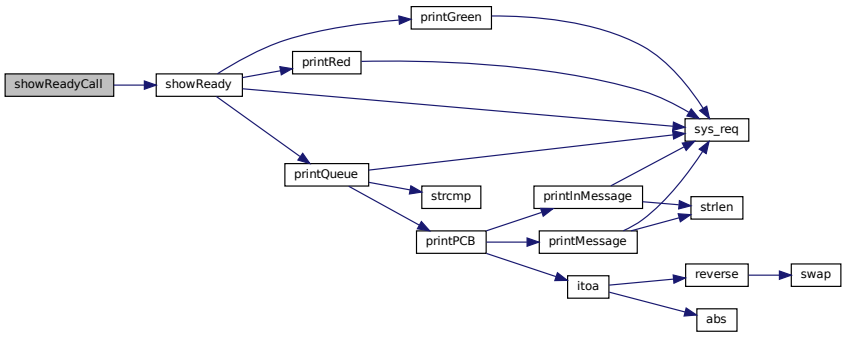
Here is the call graph for this function:



4.35.1.7 showReadyCall()

```
void showReadyCall ( )
```

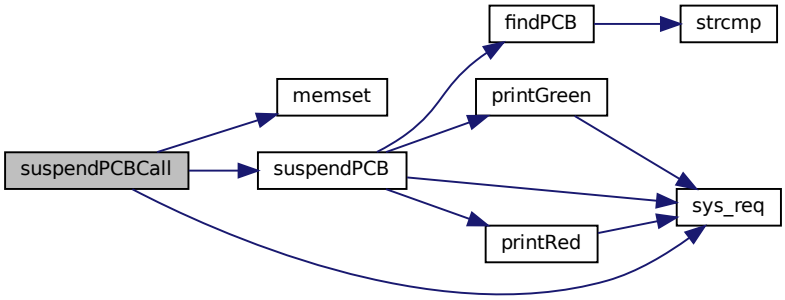
Here is the call graph for this function:



4.35.1.8 suspendPCBCall()

```
void suspendPCBCall ( )
```

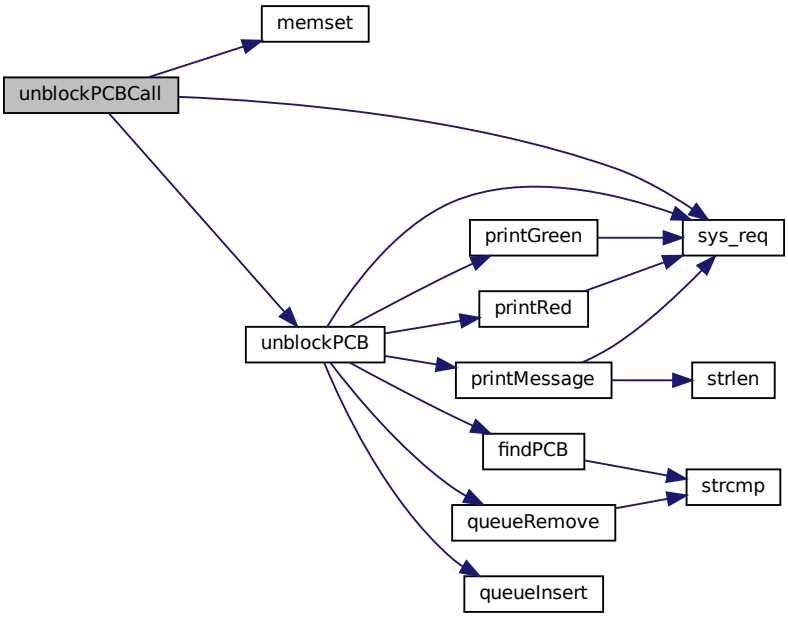
Here is the call graph for this function:



4.35.1.9 unblockPCBCall()

```
void unblockPCBCall ( )
```

Here is the call graph for this function:

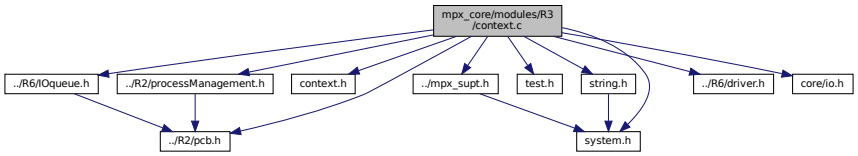


4.36 mpx_core/modules/R3/context.c File Reference

```
#include "../R2/pcb.h"
#include <system.h>
#include "context.h"
#include "../mpx_supt.h"
#include "../R2/processManagement.h"
```

```
#include "test.h"
#include "string.h"
#include "../R6/IOqueue.h"
#include "../R6/driver.h"
#include <core/io.h>
```

Include dependency graph for context.c:



Typedefs

- typedef void(∗ func_ptr) ()

Functions

- void loadProcess (char ∗name, func_ptr func)
- void loadr3 ()
- u32int ∗ sys_call (Context ∗registers)
- void yield ()
- void io_scheduler ()

Variables

- dcb_t ∗ DCB
- PCB ∗ cop
- Context ∗ old
- param params
- ioqueue_t ioqueue

4.36.1 Typedef Documentation

4.36.1.1 func_ptr

```
typedef void(* func_ptr) ()
```

4.36.2 Function Documentation

4.36.2.1 io_scheduler()

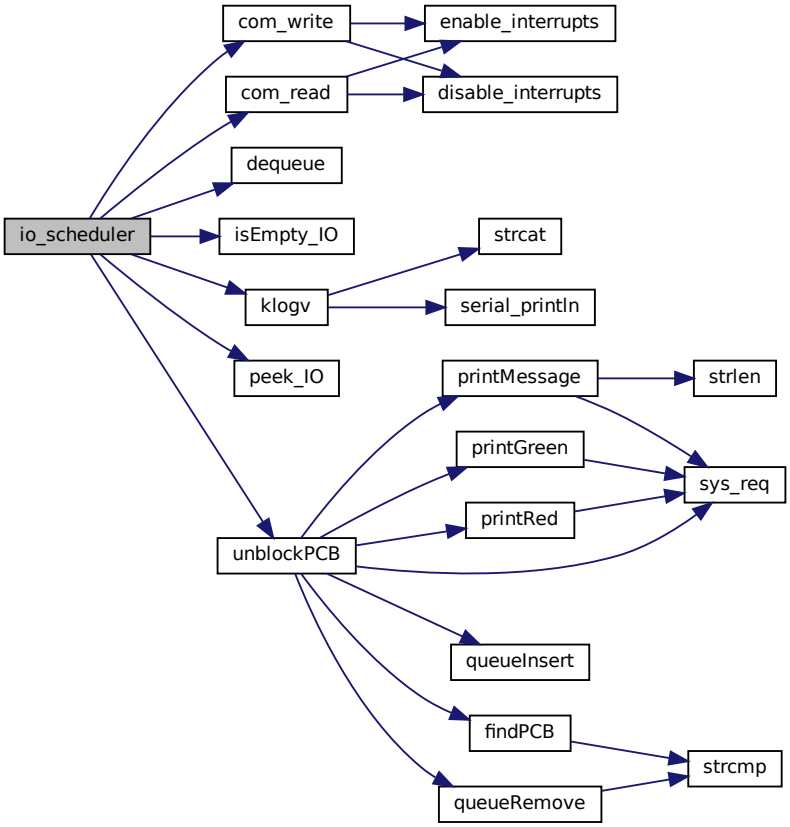
```
void io_scheduler ( )
```

+∗ io_scheduler() creates an io device for the PCB requesting IO. +∗

Parameters

(the	params you give it depends on the design of your system)

Here is the call graph for this function:



4.36.2.2 loadProcess()

```
void loadProcess (
            char * name,
            func_ptr func )
```

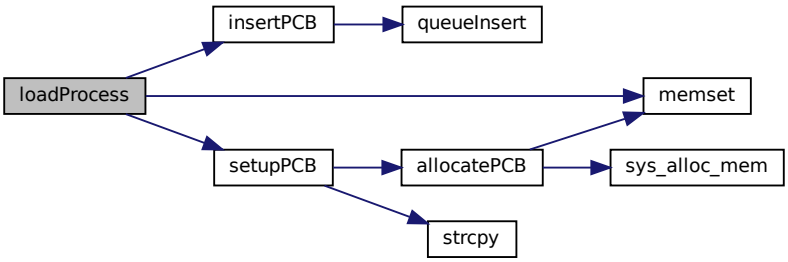
4.36.2.3 Function: loadProcess

Loads process into PCB.

Author

Farhan Shahbaz

Here is the call graph for this function:



4.36.2.4 loadr3()

```
void loadr3 ( )
```

4.36.2.5 Function: loadr3

Loads test processes into memory in a suspended ready state

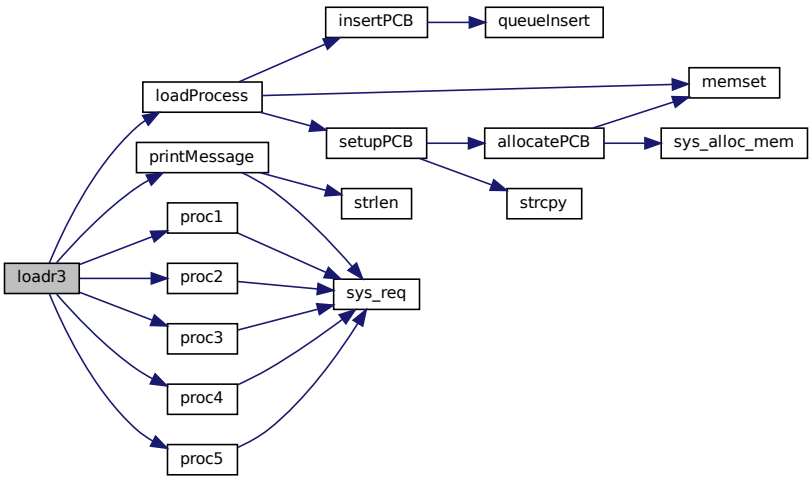
Returns

the created queue

Author

Brendan Michael, Selim Demircan

Here is the call graph for this function:



4.36.2.6 sys_call()

```
u32int* sys_call (
            Context * registers )
```

4.36.2.7 Function: sys_call

Performs context switching beteween current process and next ready process

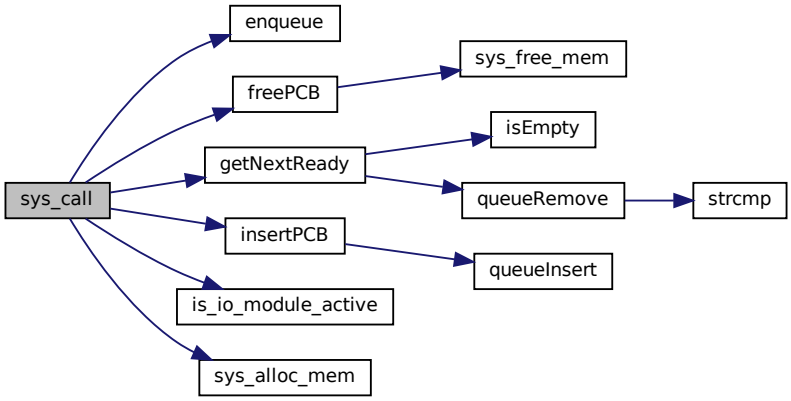
Returns

u32int*: Registers

Author

Brendan Michael, Farhan Shahbaz

Here is the call graph for this function:



4.36.2.8 yield()

```
void yield ( )
```

4.36.2.9 Function: yield

Causes the comhand to give up CPU time to other processes. Processes in the ready queue will be executed

Author

Brendan Michael

4.36.3 Variable Documentation

4.36.3.1 cop

```
PCB* cop
```

4.36.3.2 DCB

```
dcb_t* DCB  [extern]
```

4.36.3.3 ioqueue

```
ioqueue_t ioqueue  [extern]
```

4.36.3.4 old

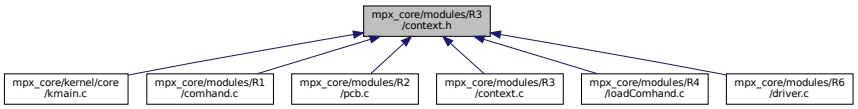
```
Context* old
```

4.36.3.5 params

```
param params  [extern]
```

4.37 mpx_core/modules/R3/context.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct Context

Typedefs

- typedef struct Context Context

Functions

- void loadr3 ()
- u32int * sys_call (Context *registers)
- void yield ()
- void io_scheduler ()

4.37.1 Typedef Documentation

4.37.1.1 Context

```
typedef struct Context Context
```

4.37.2 Function Documentation

4.37.2.1 io_scheduler()

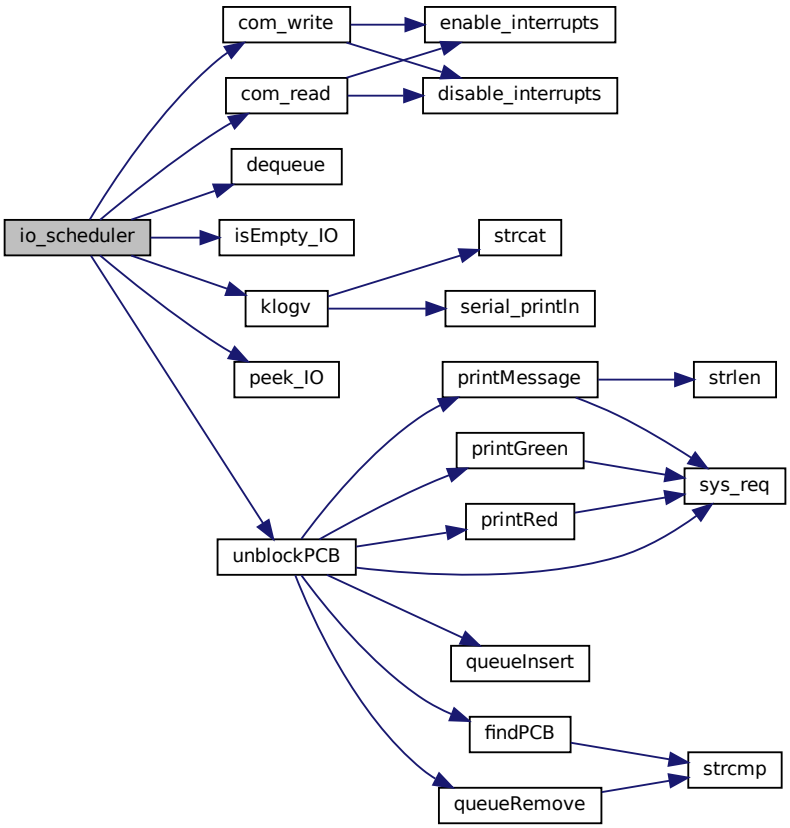
```
void io_scheduler ( )
```

+∗ io_scheduler() creates an io device for the PCB requesting IO. +∗

Parameters

(the	params you give it depends on the design of your system)

Here is the call graph for this function:



4.37.2.2 loadr3()

```
void loadr3 ( )
```

4.37.2.3 Function: loadr3

Loads test processes into memory in a suspended ready state

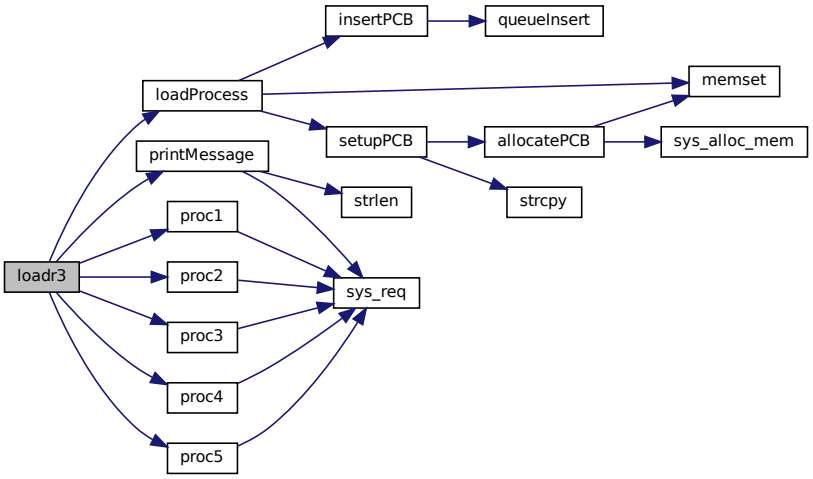
Returns

the created queue

Author

Brendan Michael, Selim Demircan

Here is the call graph for this function:



4.37.2.4 sys_call()

```
u32int* sys_call (
            Context * registers )
```

4.37.2.5 Function: sys_call

Performs context switching beteween current process and next ready process

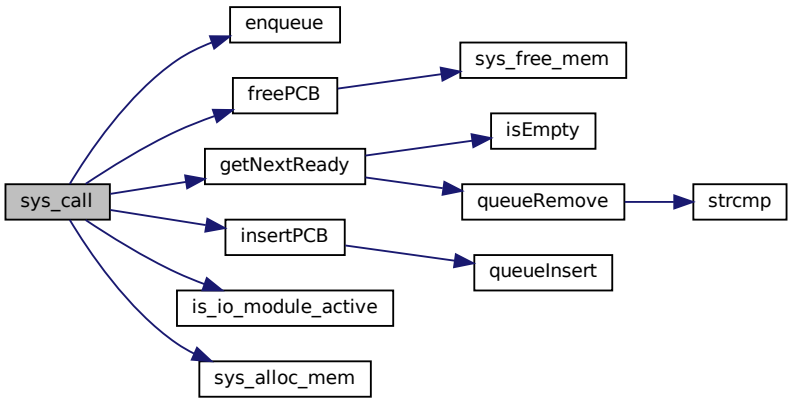
Returns

u32int*: Registers

Author

Brendan Michael, Farhan Shahbaz

Here is the call graph for this function:



4.37.2.6 yield()

```
void yield ( )
```

4.37.2.7 Function: yield

Causes the comhand to give up CPU time to other processes. Processes in the ready queue will be executed

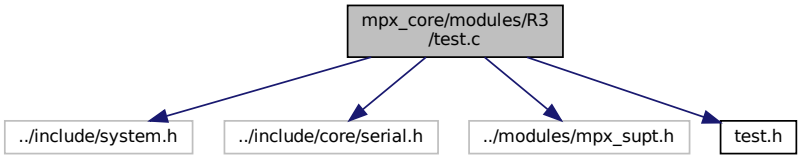
Author

Brendan Michael

4.38 mpx_core/modules/R3/test.c File Reference

```
#include "../include/system.h"
#include "../include/core/serial.h"
#include "../modules/mpx_supt.h"
#include "test.h"
```

Include dependency graph for test.c:



Macros

- #define RC_1 1
- #define RC_2 2
- #define RC_3 3
- #define RC_4 4
- #define RC_5 5

Functions

- void proc1 ()
- void proc2 ()
- void proc3 ()
- void proc4 ()
- void proc5 ()

Variables

- char * msg1 = "\nproc1 dispatched"
- char * msg2 = "\nproc2 dispatched"
- char * msg3 = "\nproc3 dispatched"
- char * msg4 = "\nproc4 dispatched"
- char * msg5 = "\nproc5 dispatched"
- int msgSize = 18
- char * er1 = "proc1 ran after it was terminated"
- char * er2 = "proc2 ran after it was terminated"
- char * er3 = "proc3 ran after it was terminated"
- char * er4 = "proc4 ran after it was terminated"
- char * er5 = "proc5 ran after it was terminated"
- int erSize = 34

4.38.1 Macro Definition Documentation

4.38.1.1 RC_1

```
#define RC_1 1
```

4.38.1.2 RC_2

```
#define RC_2 2
```

4.38.1.3 RC_3

```
#define RC_3 3
```

4.38.1.4 RC_4

```
#define RC_4 4
```

4.38.1.5 RC_5

```
#define RC_5 5
```

4.38.2 Function Documentation

4.38.2.1 proc1()

```
void proc1 ( )
```

Here is the call graph for this function:

4.38.2.2 proc2()

```
void proc2 ( )
```

Here is the call graph for this function:

4.38.2.3 proc3()

```
void proc3 ( )
```

Here is the call graph for this function:

4.38.2.4 proc4()

```
void proc4 ( )
```

Here is the call graph for this function:

4.38.2.5 proc5()

```
void proc5 ( )
```

Here is the call graph for this function:

4.38.3 Variable Documentation

4.38.3.1 er1

```
char* er1 = "proc1 ran after it was terminated"
```

4.38.3.2 er2

```
char* er2 = "proc2 ran after it was terminated"
```

4.38.3.3 er3

```
char* er3 = "proc3 ran after it was terminated"
```

4.38.3.4 er4

```
char* er4 = "proc4 ran after it was terminated"
```

4.38.3.5 er5

```
char* er5 = "proc5 ran after it was terminated"
```

4.38.3.6 erSize

```
int erSize = 34
```

4.38.3.7 msg1

```
char* msg1 = "\nproc1 dispatched"
```

4.38.3.8 msg2

```
char* msg2 = "\nproc2 dispatched"
```

4.38.3.9 msg3

```
char* msg3 = "\nproc3 dispatched"
```

4.38.3.10 msg4

```
char* msg4 = "\nproc4 dispatched"
```

4.38.3.11 msg5

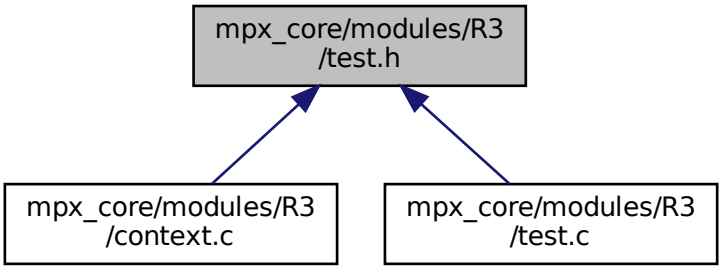
```
char* msg5 = "\nproc5 dispatched"
```

4.38.3.12 msgSize

```
int msgSize = 18
```

4.39 mpx_core/modules/R3/test.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void proc1 ()
- void proc2 ()

- void proc3 ()
- void proc4 ()
- void proc5 ()

4.39.1 Function Documentation

4.39.1.1 proc1()

```
void proc1 ( )
```

Here is the call graph for this function:

4.39.1.2 proc2()

```
void proc2 ( )
```

Here is the call graph for this function:

4.39.1.3 proc3()

```
void proc3 ( )
```

Here is the call graph for this function:

4.39.1.4 proc4()

```
void proc4 ( )
```

Here is the call graph for this function:

4.39.1.5 proc5()

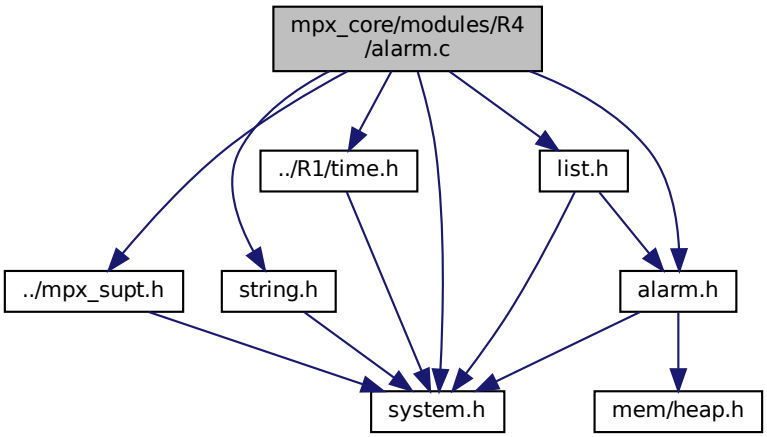
```
void proc5 ( )
```

Here is the call graph for this function:

4.40 mpx_core/modules/R4/alarm.c File Reference

```
#include "alarm.h"
#include <system.h>
#include "../mpx_supt.h"
#include <string.h>
#include "../R1/time.h"
#include "list.h"
```

Include dependency graph for alarm.c:



Functions

- void createAList ()
- void timeDaemon ()
- void createAlarmCall ()
- void createAlarm (int hh, int mm, int ss, char ∗message)
- void infiniteProcess ()

Variables

- aList ∗ head_alarms

4.40.1 Function Documentation

4.40.1.1 createAlarm()

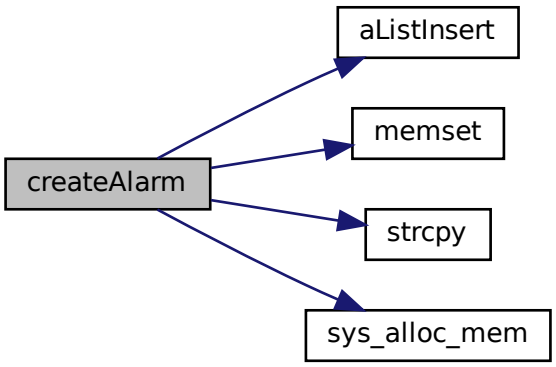
```
void createAlarm (
            int hh,
            int mm,
            int ss,
            char * message )
```

4.40.1.2 Function: createAlarm

Author

Brendan Michael

Here is the call graph for this function:



4.40.1.3 createAlarmCall()

```
void createAlarmCall ( )
```

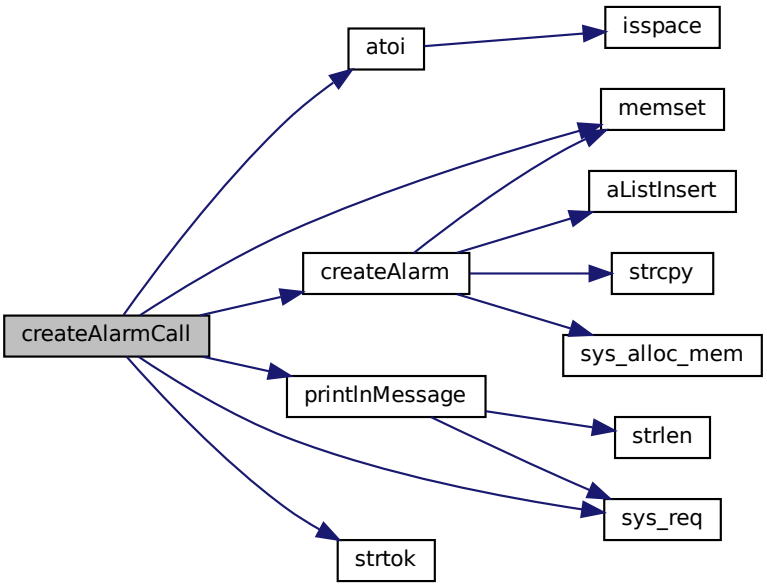
4.40.1.4 Function: createAlarmCall

Call to createAlarm that prompts user to enter alarm message/time. Sent to commhand.

Author

Farhan Shahbaz

Here is the call graph for this function:



4.40.1.5 createAList()

```
void createAList ( )
```

4.40.1.6 Function: createAlarm

Creates a new alarm based on user specified time and adds it to the list of alarms

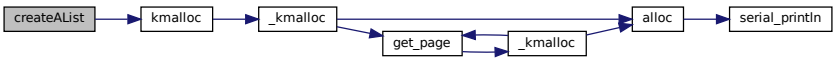
Parameters

message	the message to be displayed at the given time

Author

Brendan and Farhan

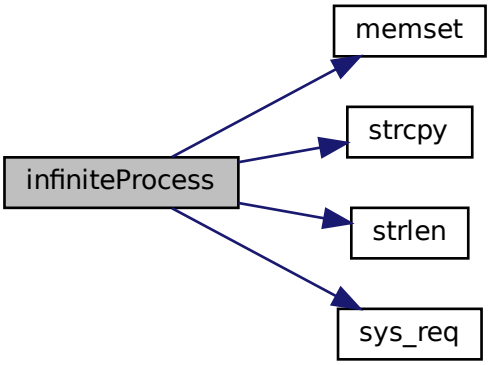
Here is the call graph for this function:



4.40.1.7 infiniteProcess()

```
void infiniteProcess ( )
```

Here is the call graph for this function:



4.40.1.8 timeDaemon()

```
void timeDaemon ( )
```

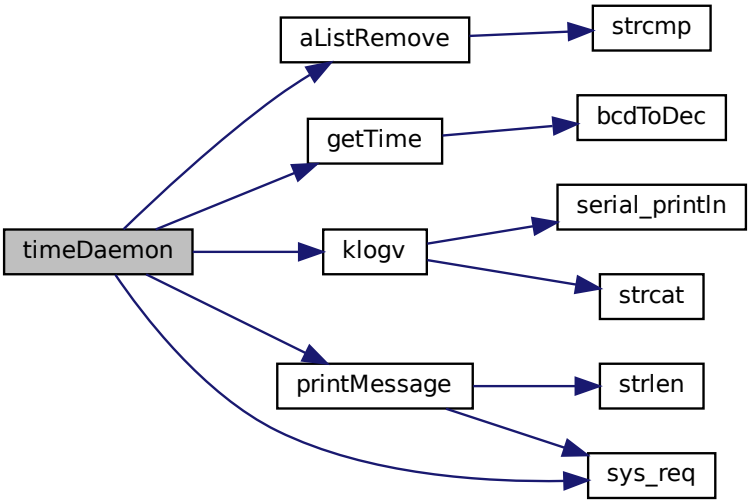
4.40.1.9 Function: timeDaemon

Background process running to check if an alarm has been set.

Author

Bryce Williams

Here is the call graph for this function:



4.40.2 Variable Documentation

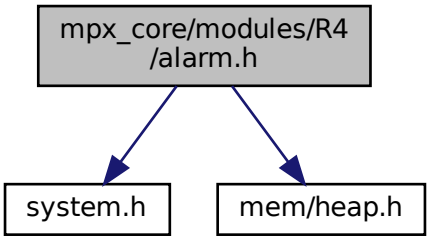
4.40.2.1 head_alarms

```
aList* head_alarms
```

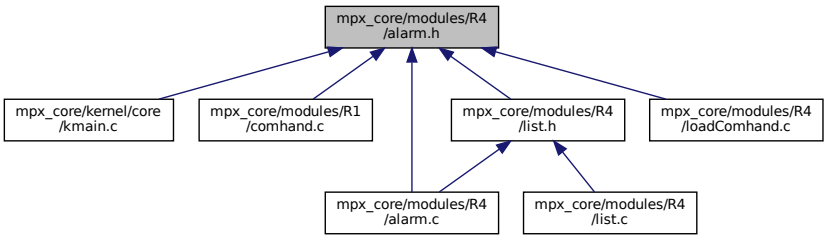
4.41 mpx_core/modules/R4/alarm.h File Reference

```
#include <system.h>
#include <mem/heap.h>
```

Include dependency graph for alarm.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct alarm_s

Typedefs

- typedef struct alarm_s alarm_t

Functions

- void createAlarmCall ()
- void createAList ()
- void createAlarm (int hh, int mm, int ss, char *message)

- void timeDaemon ()
- void infiniteProcess ()

4.41.1 Typedef Documentation

4.41.1.1 alarm_t

```
typedef struct alarm_s alarm_t
```

4.41.2 Function Documentation

4.41.2.1 createAlarm()

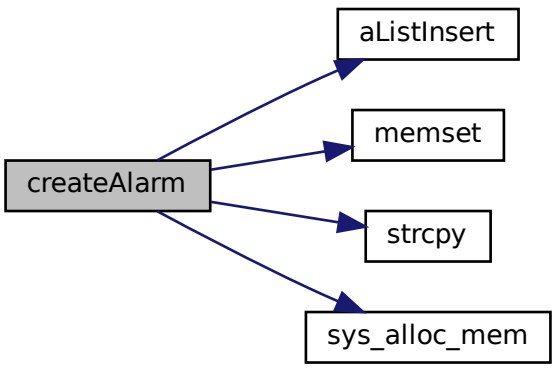
```
void createAlarm (
            int hh,
            int mm,
            int ss,
            char * message )
```

4.41.2.2 Function: createAlarm

Author

Brendan Michael

Here is the call graph for this function:



4.41.2.3 createAlarmCall()

```
void createAlarmCall ( )
```

4.41.2.4 Function: alarmCommand

Gathers and error checks user input to create an alarm.

Author

Bryce Williams

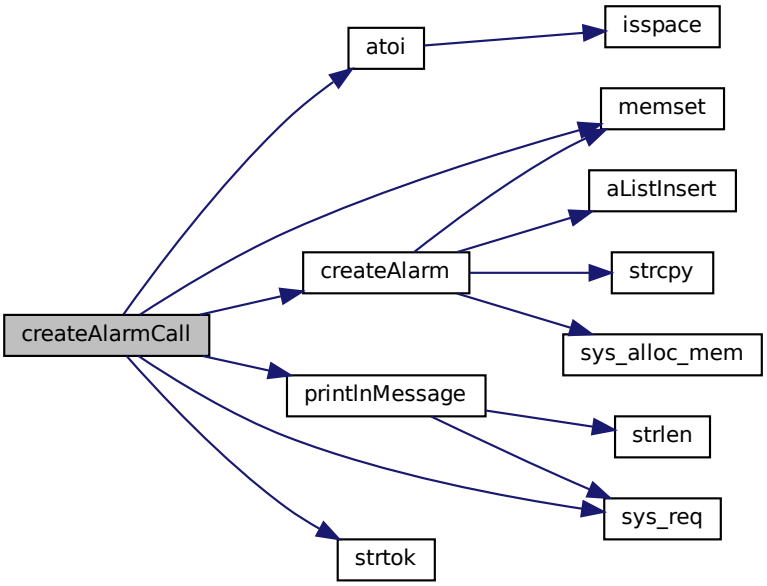
4.41.2.5 Function: createAlarmCall

Call to createAlarm that prompts user to enter alarm message/time. Sent to commhand.

Author

Farhan Shahbaz

Here is the call graph for this function:



4.41.2.6 createAList()

```
void createAList ( )
```

4.41.2.7 Function: createAlarm

Creates a new alarm based on user specified time and adds it to the list of alarms

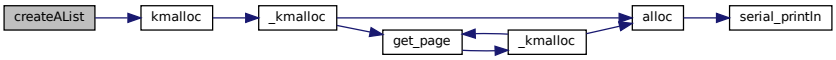
Parameters

message	the message to be displayed at the given time

Author

Brendan and Farhan

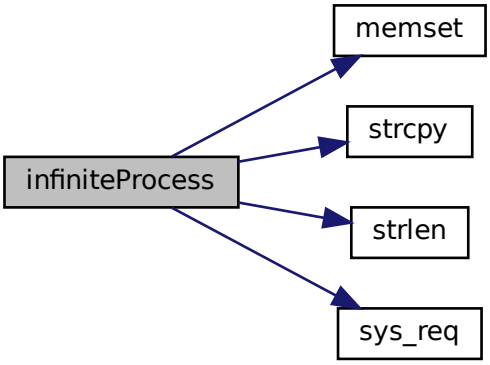
Here is the call graph for this function:



4.41.2.8 infiniteProcess()

```
void infiniteProcess ( )
```

Here is the call graph for this function:



4.41.2.9 timeDaemon()

```
void timeDaemon ( )
```

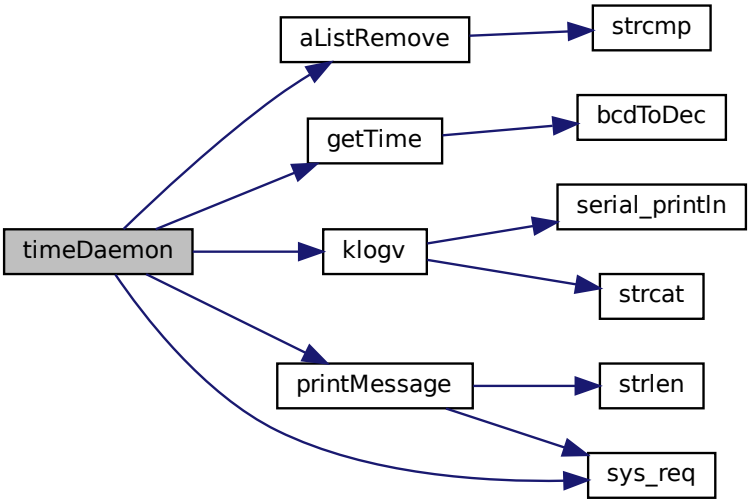
4.41.2.10 Function: timeDaemon

Background process running to check if an alarm has been set.

Author

Bryce Williams

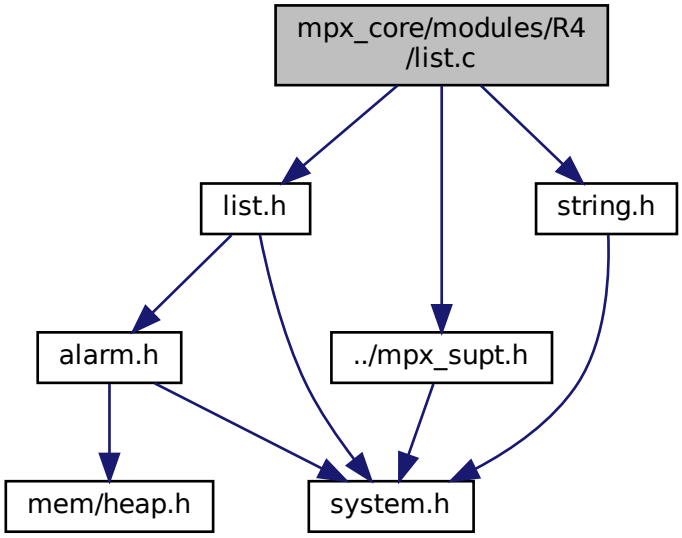
Here is the call graph for this function:



4.42 mpx_core/modules/R4/list.c File Reference

```
#include "list.h"
#include "../mpx_supt.h"
#include <string.h>
```

Include dependency graph for list.c:



Functions

- int aListInsert (aList ∗q, alarm_t ∗pcb)
- int aListRemove (aList ∗q, alarm_t ∗pcb)

4.42.1 Function Documentation

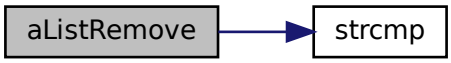
4.42.1.1 aListInsert()

```
int aListInsert (
            aList * q,
            alarm_t * pcb )
```

4.42.1.2 aListRemove()

```
int aListRemove (
            aList * q,
            alarm_t * pcb )
```

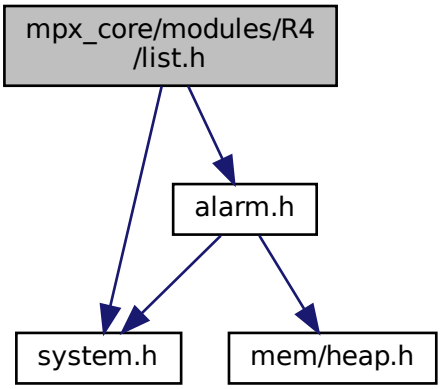
Here is the call graph for this function:



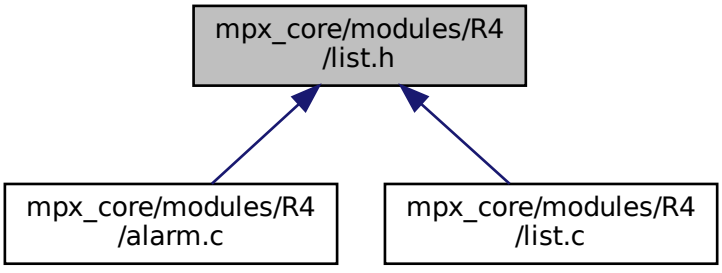
4.43 mpx_core/modules/R4/list.h File Reference

```
#include "alarm.h"
#include <system.h>
```

Include dependency graph for list.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct aList

Typedefs

- typedef struct aList aList

Functions

- int aListInsert (aList *q, alarm_t *pcb)
- int aListRemove (aList *q, alarm_t *pcb)

4.43.1 Typedef Documentation

4.43.1.1 aList

```
typedef struct aList aList
```

4.43.2 Function Documentation

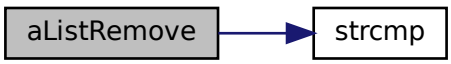
4.43.2.1 aListInsert()

```
int aListInsert (
            aList * q,
            alarm_t * pcb )
```

4.43.2.2 aListRemove()

```
int aListRemove (
            aList * q,
            alarm_t * pcb )
```

Here is the call graph for this function:

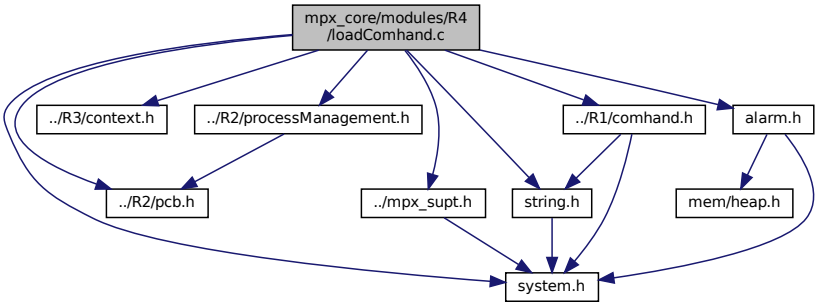


4.44 mpx_core/modules/R4/loadComhand.c File Reference

```
#include "../R2/pcb.h"
#include <system.h>
#include "../R3/context.h"
#include "../mpx_supt.h"
#include "../R2/processManagement.h"
#include "string.h"
#include "../R1/comhand.h"
```

```
#include "alarm.h"
```

Include dependency graph for loadComhand.c:



Functions

- void loadComhand ()
- void loadIdle ()
- void loadTimeDaemon ()
- void loadInfiniteProcess ()

4.44.1 Function Documentation

4.44.1.1 loadComhand()

```
void loadComhand ( )
```

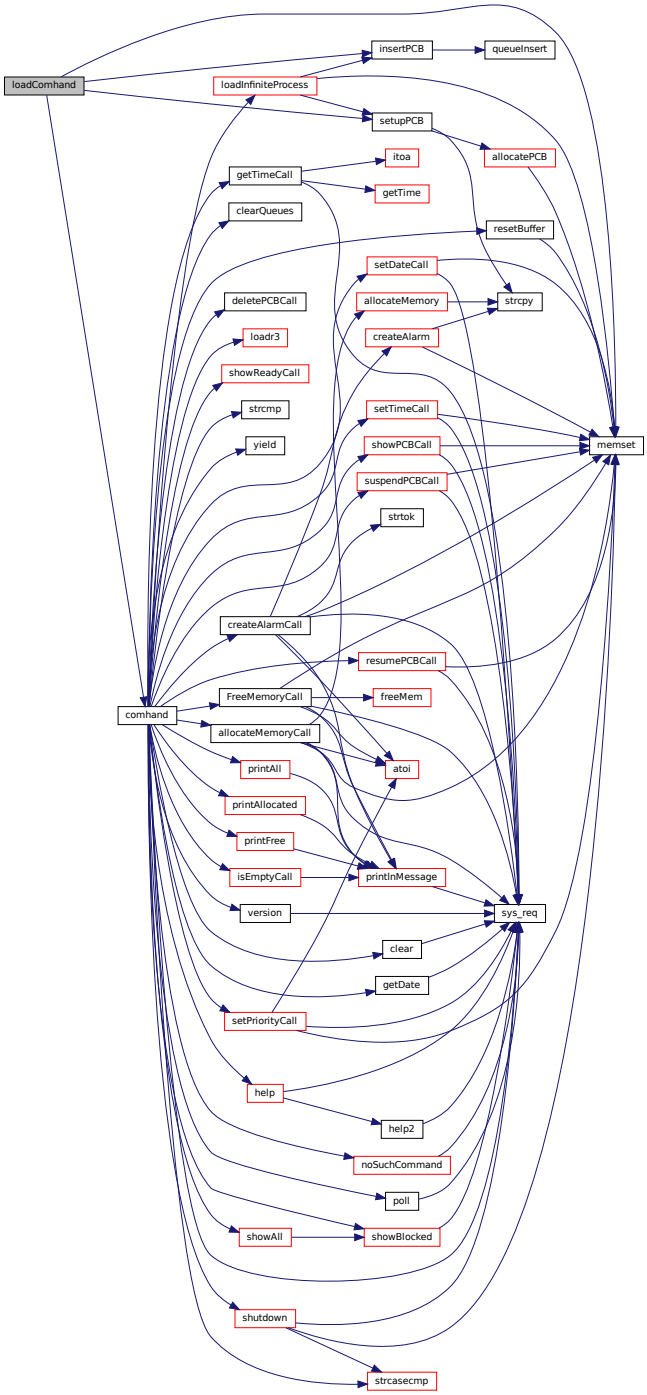
4.44.1.2 Function: loadComhand

Loads comhand as a system process with a high set priority.

Author

Bryce Williams

Here is the call graph for this function:



4.44.1.3 loadIdle()

```
void loadIdle ( )
```

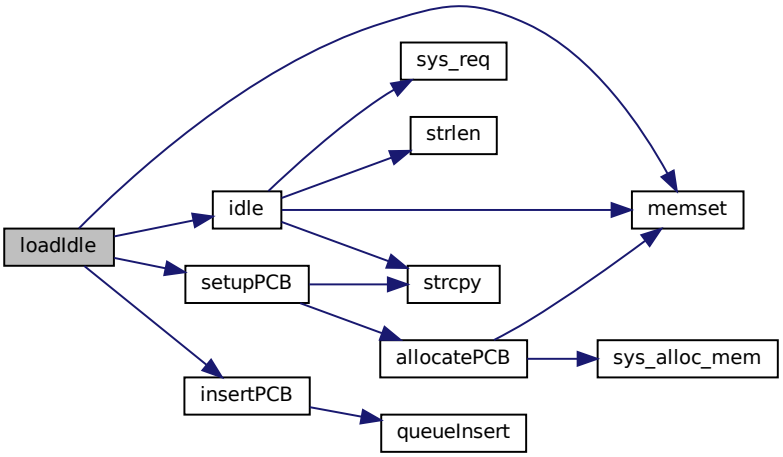
4.44.1.4 Function: loadIdle

Loads idle as a system process with a low set piority

Author

Bryce Williams

Here is the call graph for this function:



4.44.1.5 loadInfiniteProcess()

```
void loadInfiniteProcess ( )
```

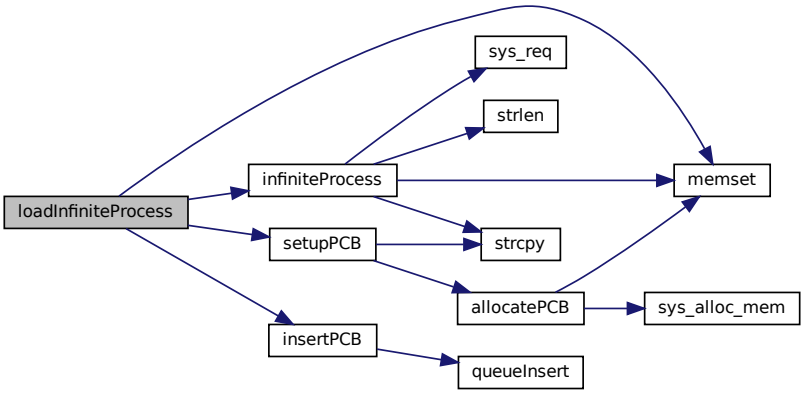
4.44.1.6 Function: loadInfiniteProcess

Loads an infinite process as an application with a low set priority.

Author

Brendan Michael

Here is the call graph for this function:



4.44.1.7 loadTimeDaemon()

```
void loadTimeDaemon ( )
```

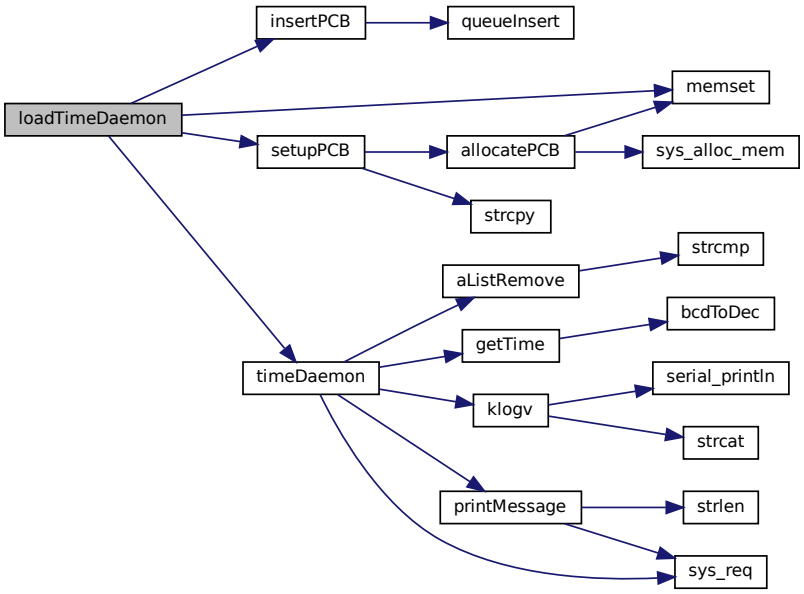
4.44.1.8 Function: loadTimeDaemon

Loads TimeDaemon as a system process with a high set priority.

Author

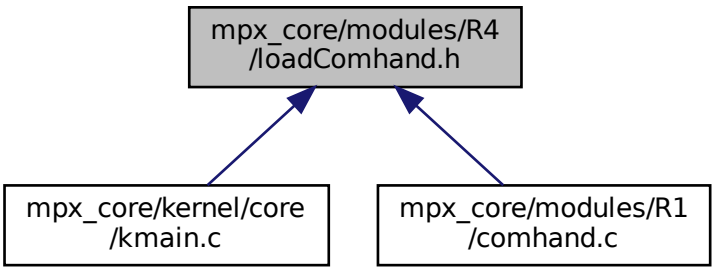
Bryce Williams

Here is the call graph for this function:



4.45 mpx_core/modules/R4/loadComhand.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void loadComhand ()
- void loadIdle ()
- void loadTimeDaemon ()
- void loadInfiniteProcess ()

4.45.1 Function Documentation

4.45.1.1 loadComhand()

```
void loadComhand ( )
```

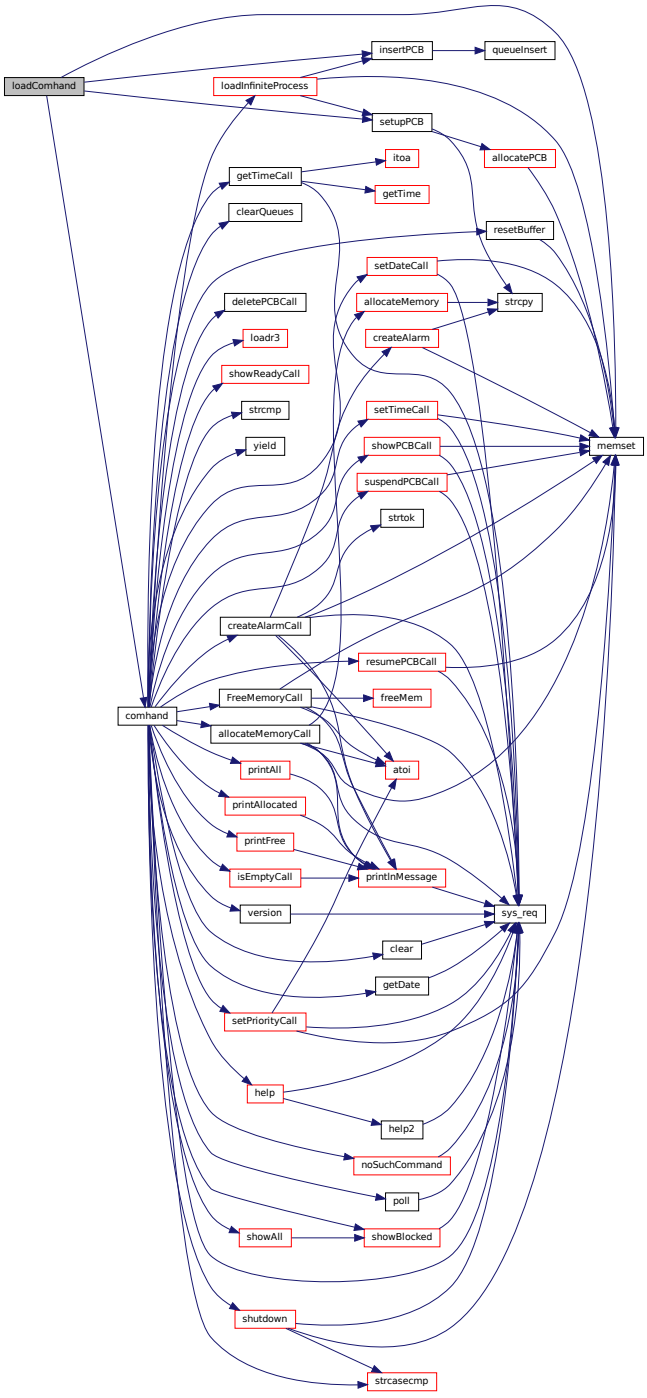
4.45.1.2 Function: loadComhand

Loads comhand as a system process with a high set priority.

Author

Bryce Williams

Here is the call graph for this function:



4.45.1.3 loadIdle()

```
void loadIdle ( )
```

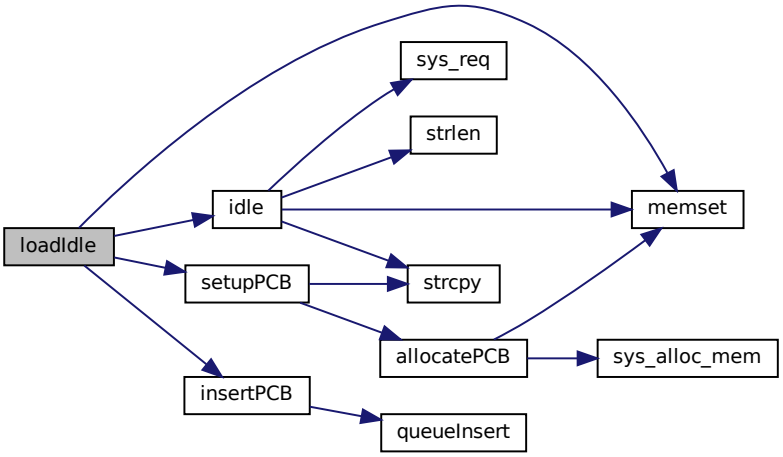
4.45.1.4 Function: loadIdle

Loads idle as a system process with a low set piority

Author

Bryce Williams

Here is the call graph for this function:



4.45.1.5 loadInfiniteProcess()

```
void loadInfiniteProcess ( )
```

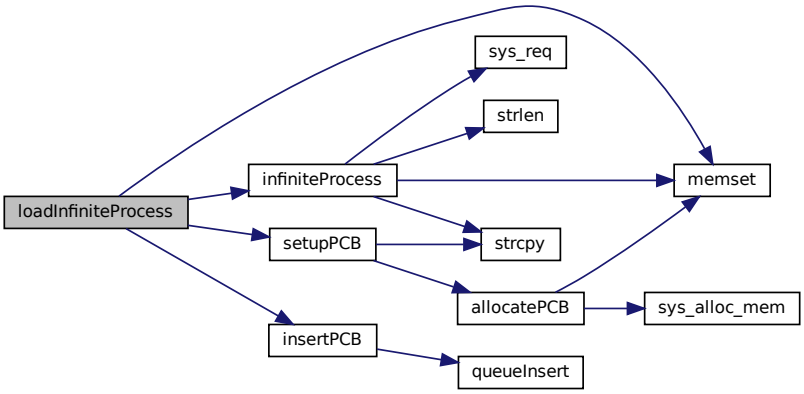
4.45.1.6 Function: loadInfiniteProcess

Loads an infinite process as an application with a low set priority.

Author

Brendan Michael

Here is the call graph for this function:



4.45.1.7 loadTimeDaemon()

```
void loadTimeDaemon ( )
```

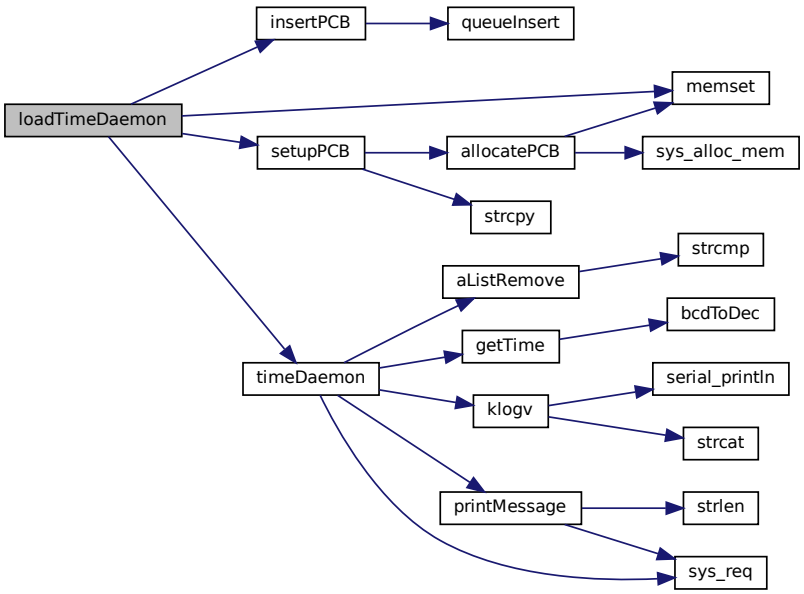
4.45.1.8 Function: loadTimeDaemon

Loads TimeDaemon as a system process with a high set priority.

Author

Bryce Williams

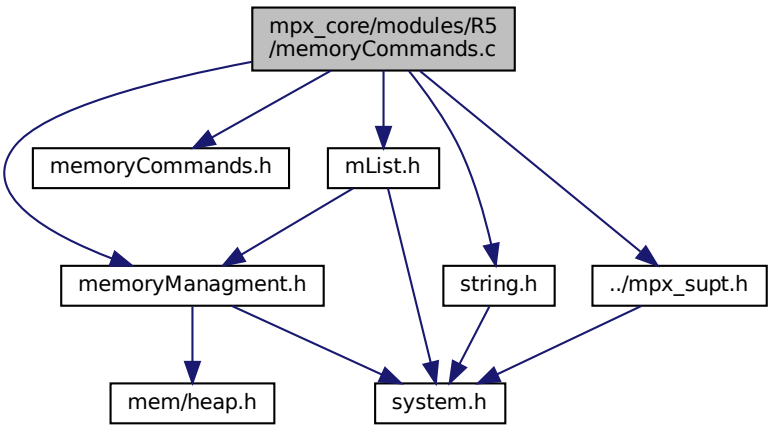
Here is the call graph for this function:



4.46 mpx_core/modules/R5/memoryCommands.c File Reference

```
#include "memoryManagment.h"
#include "memoryCommands.h"
#include "../mpx_supt.h"
#include <string.h>
#include "mList.h"
```

Include dependency graph for memoryCommands.c:



Functions

- void allocateMemoryCall ()
- void FreeMemoryCall ()
- void isEmptyCall ()

4.46.1 Function Documentation

4.46.1.1 allocateMemoryCall()

```
void allocateMemoryCall ( )
```

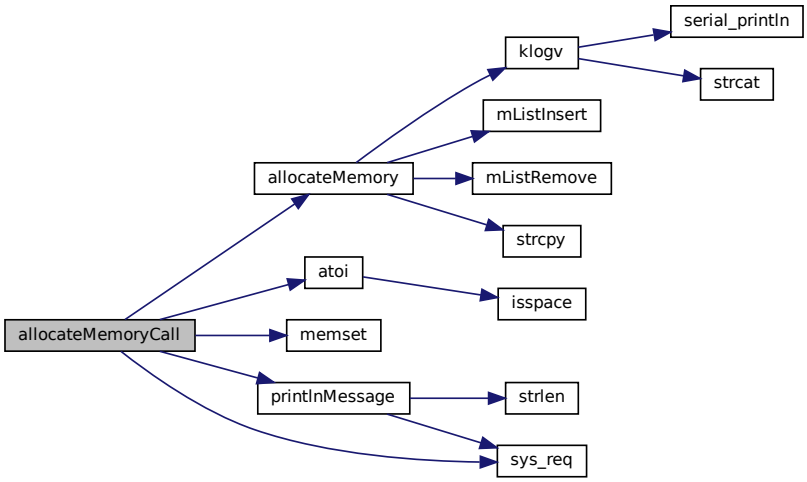
4.46.1.2 Function: allocateMemoryCall

Prompts the user for how much memory they will like to allocate, in bytes, and allocates it using the allocateMemory function(first fit)

Authors

Farhan Shahbaz

Here is the call graph for this function:



4.46.1.3 FreeMemoryCall()

```
void FreeMemoryCall ( )
```

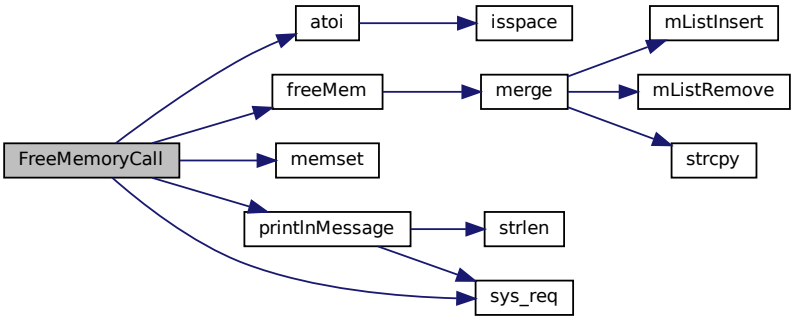
4.46.1.4 Function: freeMemoryCall

Prompts the user for which memory address they would like to free (has to be allocated), and frees it using the freeMem function

Authors

Farhan Shahbaz

Here is the call graph for this function:



4.46.1.5 isEmptyCall()

```
void isEmptyCall ( )
```

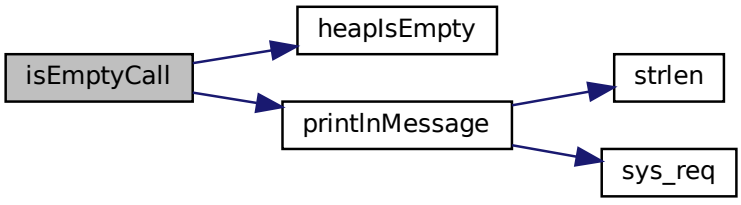
4.46.1.6 Function: isEmptyCall

Informs the user if the heap is empty

Authors

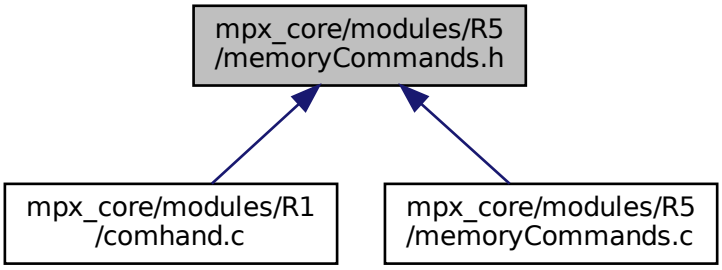
Selim Demircan

Here is the call graph for this function:



4.47 mpx_core/modules/R5/memoryCommands.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void allocateMemoryCall ()
- void FreeMemoryCall ()
- void isEmptyCall ()

4.47.1 Function Documentation

4.47.1.1 allocateMemoryCall()

```
void allocateMemoryCall ( )
```

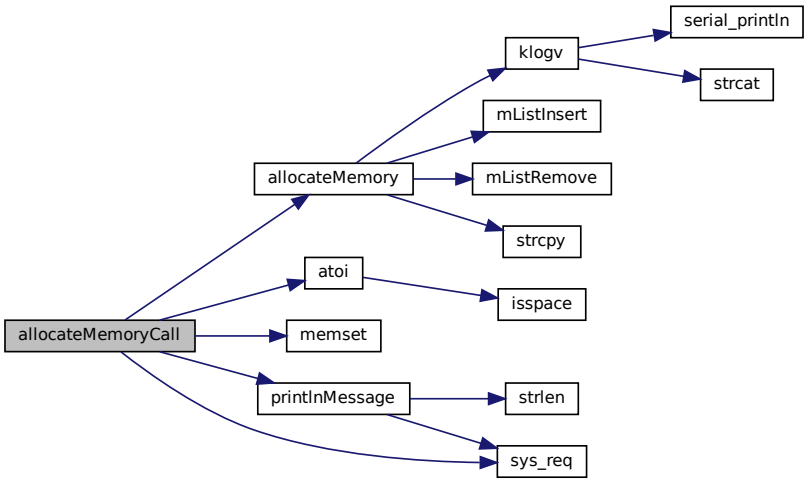
4.47.1.2 Function: allocateMemoryCall

Prompts the user for how much memory they will like to allocate, in bytes, and allocates it using the allocateMemory function(first fit)

Authors

Farhan Shahbaz

Here is the call graph for this function:



4.47.1.3 FreeMemoryCall()

```
void FreeMemoryCall ( )
```

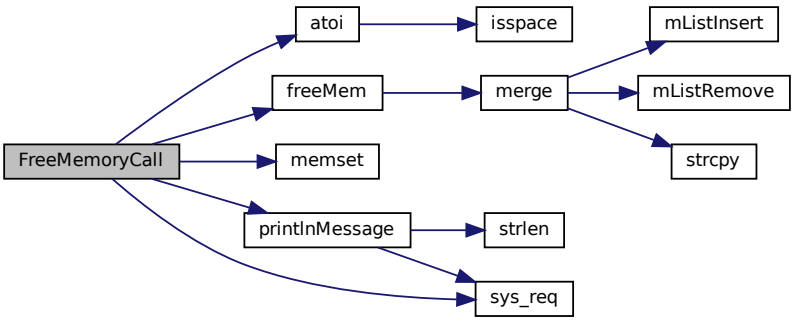
4.47.1.4 Function: freeMemoryCall

Prompts the user for which memory address they would like to free (has to be allocated), and frees it using the freeMem function

Authors

Farhan Shahbaz

Here is the call graph for this function:



4.47.1.5 isEmptyCall()

```
void isEmptyCall ( )
```

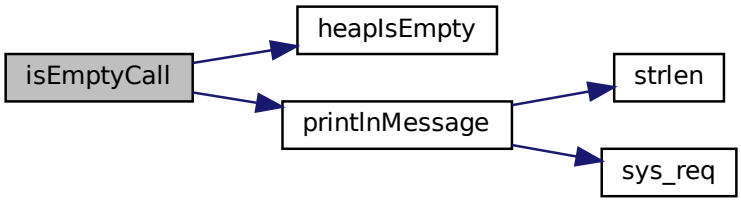
4.47.1.6 Function: isEmptyCall

Informs the user if the heap is empty

Authors

Selim Demircan

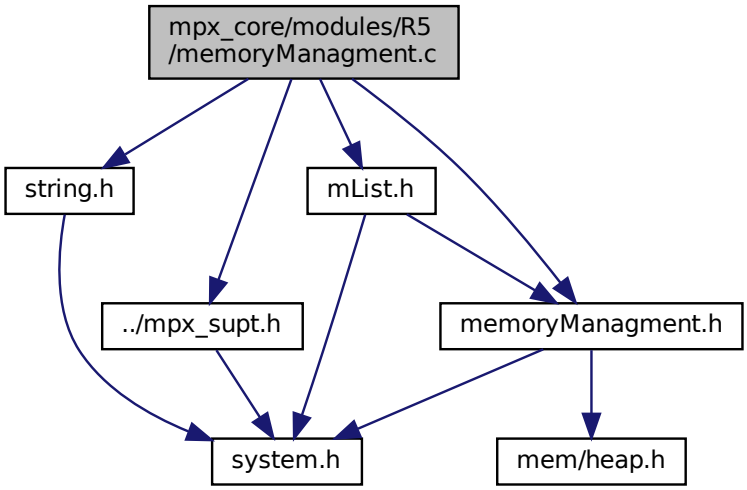
Here is the call graph for this function:



4.48 mpx_core/modules/R5/memoryManagment.c File Reference

```
#include "memoryManagment.h"
#include "../mpx_supt.h"
#include <string.h>
#include "mList.h"
```

Include dependency graph for memoryManagment.c:



Functions

- void createmList ()
- int initializeHeap (int size)
- u32int allocateMemory (u32int bytes)
- int freeMem (void *ptr)
- void merge (CMCB_t *current)
- int heapIsEmpty ()
- void printCMCB (CMCB_t *block)
- void printAllocated ()
- void printFree ()
- void printAll ()

Variables

- mList * MCBList
- void * heap_top = NULL

4.48.1 Function Documentation

4.48.1.1 allocateMemory()

```
u32int allocateMemory (
            u32int bytes )
```

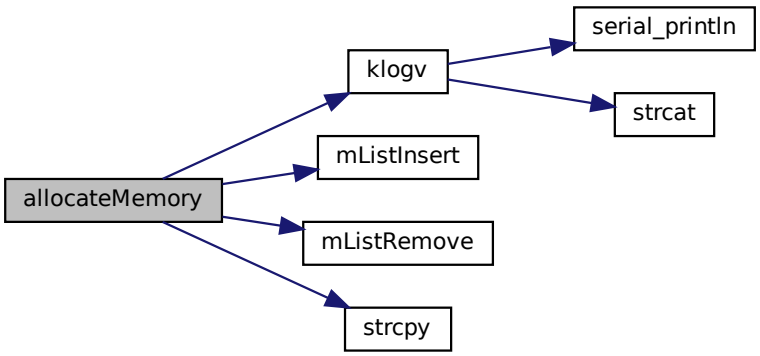
4.48.1.2 Function: allocateMemory

Allocates memory from the heap by taking in a specific amount of bytes. Uses the first fit method for the memory blocks

Authors

Brenden Michael, Farhan Shahbaz

Here is the call graph for this function:



4.48.1.3 createmList()

```
void createmList ( )
```

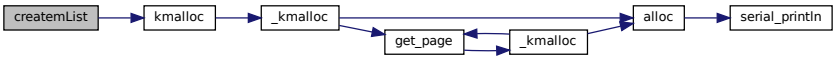
4.48.1.4 Function: createmList

Initializes MCBList. Allocates the memory, sets head and tail to null, and sets count to 0.

Authors

Brendan Michael

Here is the call graph for this function:



4.48.1.5 freeMem()

```
int freeMem (
            void * ptr )
```

4.48.1.6 Function: freeMem

Frees up memory specified by the address of ptr. Returns 0 if the address does not exist or the block is already free.Returns 1 if succesfully freed.

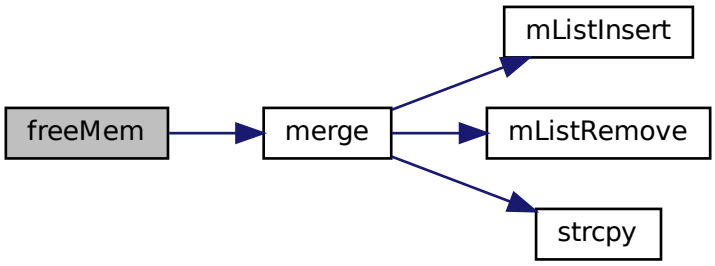
Parameters

ptr-	Pointer to memory location that needs to be freed.

Authors

Bryce Williams

Here is the call graph for this function:



4.48.1.7 heapIsEmpty()

```
int heapIsEmpty ( )
```

4.48.1.8 Function: heapIsEmpty

Checks to see if the heap is empty. Returns 0 for false and 1 for true

Authors

Farhan Shahbaz

4.48.1.9 initializeHeap()

```
int initializeHeap (
            int size )
```

4.48.1.10 Function: initializeHeap

Allocates the heap memory specified by the parameter size using kmalloc.

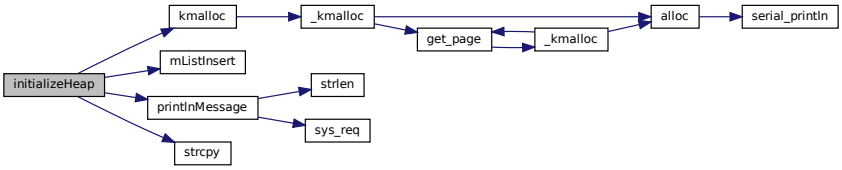
Parameters

size-	The size of the heap in bytes.

Authors

Bryce Williams

Here is the call graph for this function:



4.48.1.11 merge()

```
void merge (
            CMCB_t * current )
```

4.48.1.12 Function: merge

Merges the memory specified by current with the rest of the free memory if possible.

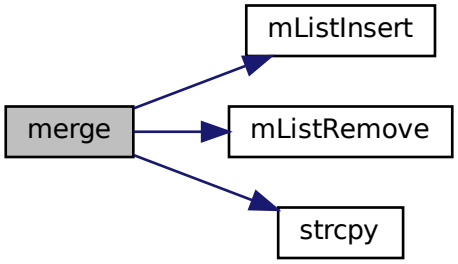
Parameters

current-	pointer to the memory address that needs to be merged.

Authors

Brendan Michael

Here is the call graph for this function:



4.48.1.13 printAll()

```
void printAll ( )
```

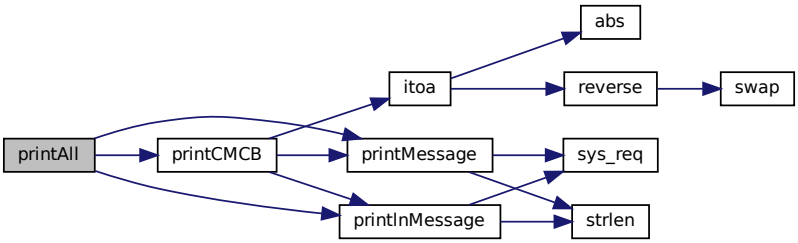
4.48.1.14 Function: printAll

Prints allocated and free blocks of memory.

Authors

Selim Demircan

Here is the call graph for this function:



4.48.1.15 printAllocated()

```
void printAllocated ( )
```

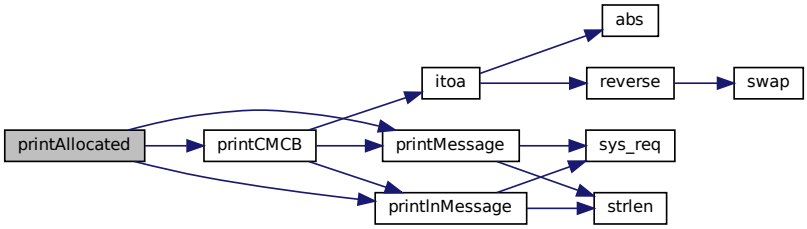
4.48.1.16 Function: printAllocated

Prints all allocated blocks of memory.

Authors

Selim Demircan

Here is the call graph for this function:



4.48.1.17 printCMCB()

```
void printCMCB (
            CMCB_t * block )
```

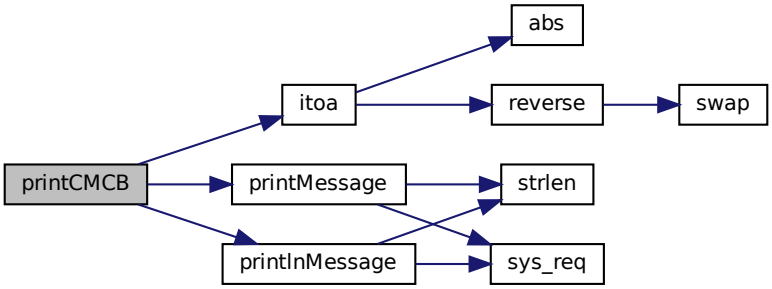
4.48.1.18 Function: printCMCB

Prints the block information and block address. Takes the pointer to the memory block.

Authors

Selim Demircan

Here is the call graph for this function:



4.48.1.19 printFree()

```
void printFree ( )
```

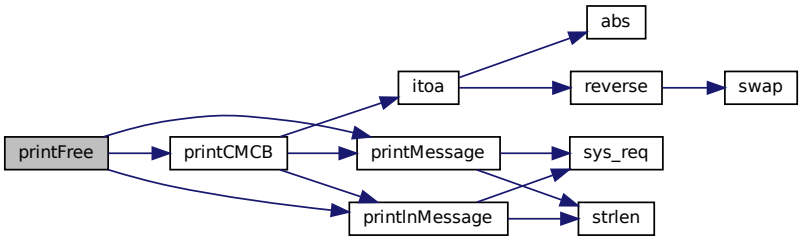
4.48.1.20 Function: printFree

Prints all free blocks of memory.

Authors

Selim Demircan

Here is the call graph for this function:



4.48.2 Variable Documentation

4.48.2.1 heap_top

```
void* heap_top = NULL
```

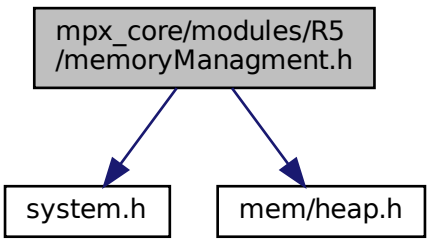
4.48.2.2 MCBList

```
mList* MCBList
```

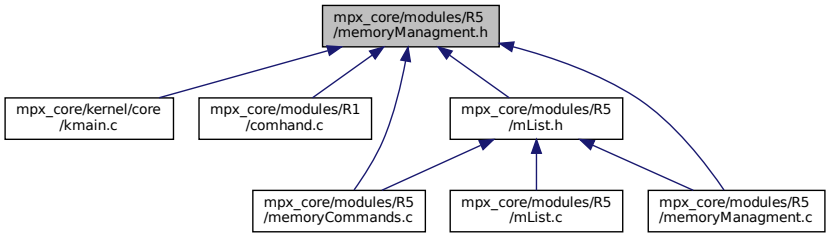
4.49 mpx_core/modules/R5/memoryManagment.h File Reference

```
#include <system.h>
#include <mem/heap.h>
```

Include dependency graph for memoryManagment.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct CMCB_s
- struct LMCB_s

Macros

- #define HEAP_SIZE 50000
- #define FREE_MEMORY 1
- #define ALLOCATED_MEMORY 2

Typedefs

- typedef struct CMCB_s CMCB_t
- typedef struct LMCB_s LMCB_t

Functions

- int heapIsEmpty ()
- int initializeHeap (int size)
- void merge (CMCB_t *current)
- u32int allocateMemory (u32int bytes)
- int freeMem (void *ptr)
- void printCMCB (CMCB_t *block)
- void printAllocated ()
- void printFree ()
- void createmList ()
- void printAll ()

4.49.1 Macro Definition Documentation

4.49.1.1 ALLOCATED_MEMORY

```
#define ALLOCATED_MEMORY 2
```

4.49.1.2 FREE_MEMORY

```
#define FREE_MEMORY 1
```

4.49.1.3 HEAP_SIZE

```
#define HEAP_SIZE 50000
```

4.49.2 Typedef Documentation

4.49.2.1 CMCB_t

```
typedef struct CMCB_s CMCB_t
```

4.49.2.2 LMCB_t

```
typedef struct LMCB_s LMCB_t
```

4.49.3 Function Documentation

4.49.3.1 allocateMemory()

```
u32int allocateMemory (
            u32int bytes )
```

4.49.3.2 Function: allocateMemory

Allocates a memory block of the desired size. Uses the first fit strategy, the first first free memory block with a large enough size is the one allocated.

Parameters

u32int	bytes: The requested block size

Returns

u32int: returns the address of the allocated block

Author

Brendan Michael

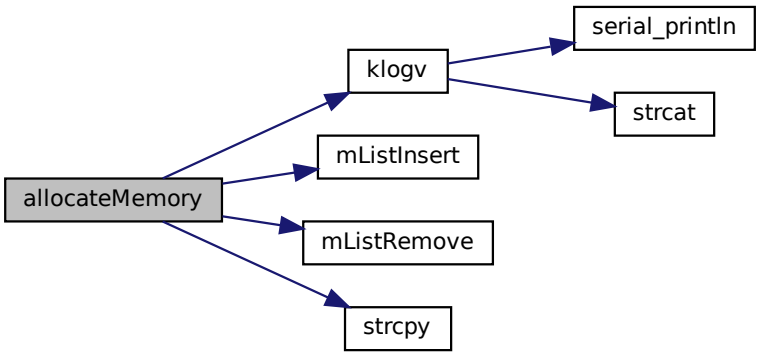
4.49.3.3 Function: allocateMemory

Allocates memory from the heap by taking in a specific amount of bytes. Uses the first fit method for the memory blocks

Authors

Brenden Michael, Farhan Shahbaz

Here is the call graph for this function:



4.49.3.4 createmList()

```
void createmList ( )
```

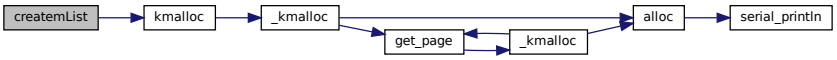
4.49.3.5 Function: createmList

Initializes MCBList. Allocates the memory, sets head and tail to null, and sets count to 0.

Authors

Brendan Michael

Here is the call graph for this function:



4.49.3.6 freeMem()

```
int freeMem (
            void * ptr )
```

4.49.3.7 Function: freeMem

Frees the memory at a given address. Calls merge to combine the freed block with adjacent free blocks.

Parameters

void	∗ptr: Memory address of block to be freed

Returns

int: returns 1 is sucessful 0 otherwise

Author

Brendan Michael

4.49.3.8 Function: freeMem

Frees up memory specified by the address of ptr. Returns 0 if the address does not exist or the block is already free.Returns 1 if succesfully freed.

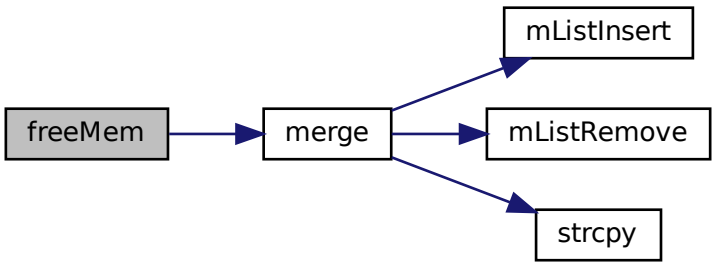
Parameters

ptr-	Pointer to memory location that needs to be freed.

Authors

Bryce Williams

Here is the call graph for this function:



4.49.3.9 heapIsEmpty()

```
int heapIsEmpty ( )
```

4.49.3.10 Function: heapIsEmpty

Searches the memory list and returns 1 if no allocated blocks are found in it. @ param int size: size of desired heap @ return int: returns 1 if empty 0 otherwise

Author

Brendan Michael

4.49.3.11 Function: heapIsEmpty

Checks to see if the heap is empty. Returns 0 for false and 1 for true

Authors

Farhan Shahbaz

4.49.3.12 initializeHeap()

```
int initializeHeap (
            int size )
```

4.49.3.13 Function: initializeHeap

Initializes the heap that stores memory blocks. Inserts a single CMCB that represents the free memory allocated for the heap. @ param int size: size of desired heap @ return int: returns 1 if sucessful 0 otherwise

Author

Brendan Michael

4.49.3.14 Function: initializeHeap

Allocates the heap memory specified by the parameter size using kmalloc.

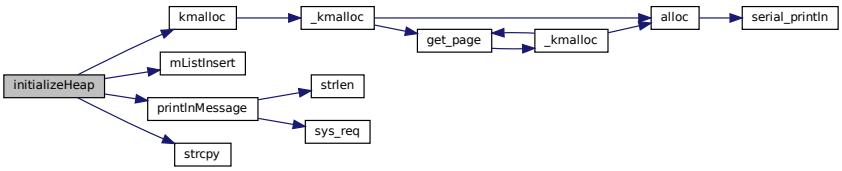
Parameters

size-	The size of the heap in bytes.

Authors

Bryce Williams

Here is the call graph for this function:



4.49.3.15 merge()

```
void merge (
            CMCB_t * current )
```

4.49.3.16 Function: merge

Merges the just freed memory block with adjacent free memory blocks. Merged blocks are removed from the memory list. @ param CMCB_t ∗current: the just freed memory block to be merged

Author

Brendan Michael

4.49.3.17 Function: merge

Merges the memory specified by current with the rest of the free memory if possible.

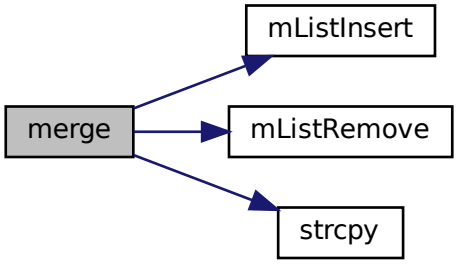
Parameters

current-	pointer to the memory address that needs to be merged.

Authors

Brendan Michael

Here is the call graph for this function:



4.49.3.18 printAll()

```
void printAll ( )
```

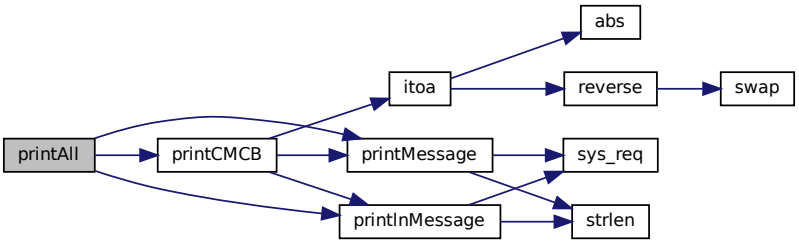
4.49.3.19 Function: printAll

Prints allocated and free blocks of memory.

Authors

Selim Demircan

Here is the call graph for this function:



4.49.3.20 printAllocated()

```
void printAllocated ( )
```

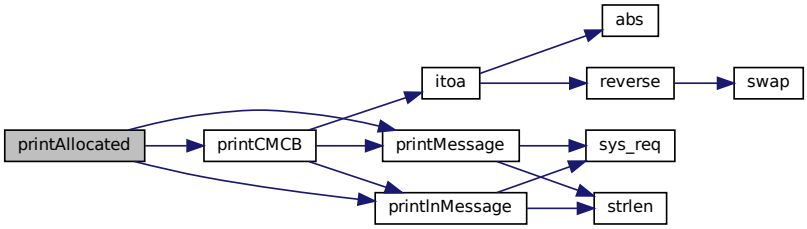
4.49.3.21 Function: printAllocated

Prints all allocated blocks of memory.

Authors

Selim Demircan

Here is the call graph for this function:



4.49.3.22 printCMCB()

```
void printCMCB (
            CMCB_t * block )
```

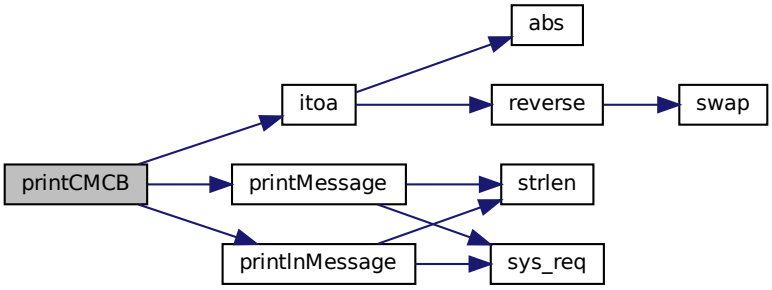
4.49.3.23 Function: printCMCB

Prints the block information and block address. Takes the pointer to the memory block.

Authors

Selim Demircan

Here is the call graph for this function:



4.49.3.24 printFree()

```
void printFree ( )
```

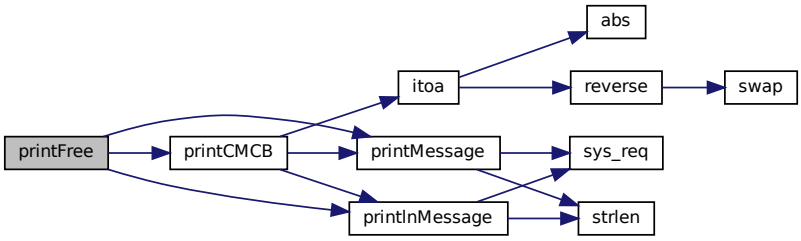
4.49.3.25 Function: printFree

Prints all free blocks of memory.

Authors

Selim Demircan

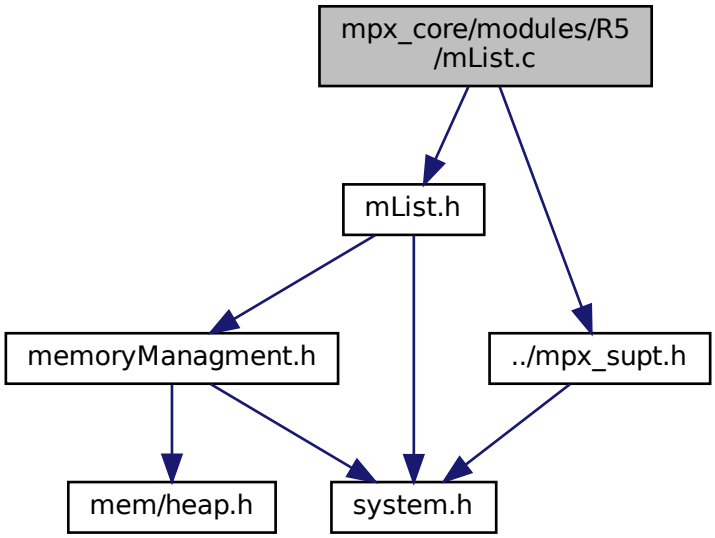
Here is the call graph for this function:



4.50 mpx_core/modules/R5/mList.c File Reference

```
#include "mList.h"
#include "../mpx_supt.h"
```

Include dependency graph for mList.c:



Functions

- int mListInsert (mList *q, CMCB_t *cmcb)
- int mListRemove (mList *q, CMCB_t *cmcb)

4.50.1 Function Documentation

4.50.1.1 mListInsert()

```
int mListInsert (
            mList * q,
            CMCB_t * cmcb )
```

4.50.1.2 Function: mListInsert

Inserts cmcb block into the list specified by q.

Parameters

q-Pointer	to the list.
cmcb-	Pointer to the cmcb block to be inserted into the list.

Authors

Bryce Williams

4.50.1.3 mListRemove()

```
int mListRemove (
            mList * q,
            CMCB_t * cmcb )
```

4.50.1.4 Function: mListInsert

Removes cmcb block from the list specified by q.

Parameters

q-Pointer	to the list.
cmcb-	Pointer to the cmcb block to be removed from the list.

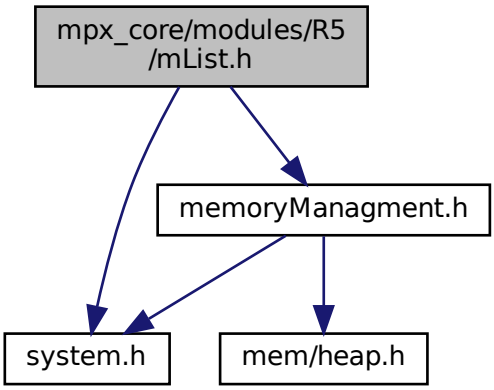
Authors

Bryce Williams

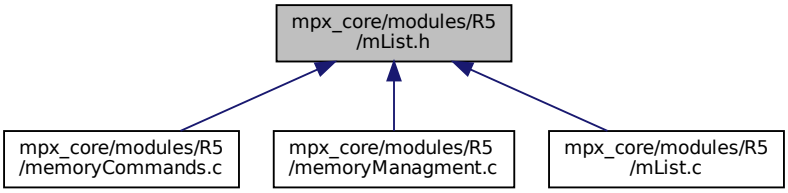
4.51 mpx_core/modules/R5/mList.h File Reference

```
#include "memoryManagment.h"
#include <system.h>
```

Include dependency graph for mList.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct mList

Typedefs

- typedef struct mList mList

Functions

- int mListInsert (mList *q, CMCB_t *pcb)
- int mListRemove (mList *q, CMCB_t *pcb)

4.51.1 Typedef Documentation

4.51.1.1 mList

```
typedef struct mList mList
```

4.51.2 Function Documentation

4.51.2.1 mListInsert()

```
int mListInsert (
            mList * q,
            CMCB_t * cmcb )
```

4.51.2.2 Function: mListInsert

Inserts cmcb block into the list specified by q.

Parameters

q-Pointer	to the list.
cmcb-	Pointer to the cmcb block to be inserted into the list.

Authors

Bryce Williams

4.51.2.3 mListRemove()

```
int mListRemove (
            mList * q,
            CMCB_t * cmcb )
```

4.51.2.4 Function: mListInsert

Removes cmcb block from the list specified by q.

Parameters

q-Pointer	to the list.
cmcb-	Pointer to the cmcb block to be removed from the list.

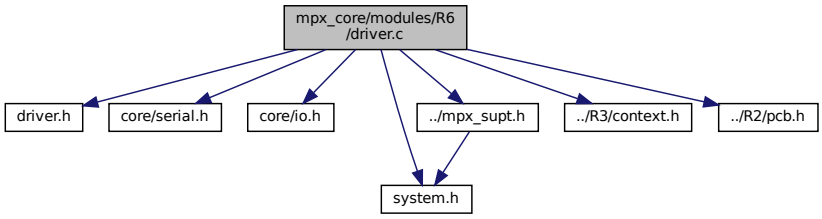
Authors

Bryce Williams

4.52 mpx_core/modules/R6/driver.c File Reference

```
#include "driver.h"
#include <core/serial.h>
#include <core/io.h>
#include "system.h"
#include "../mpx_supt.h"
#include "../R3/context.h"
#include "../R2/pcb.h"
```

Include dependency graph for driver.c:



Functions

- void disable_interrupts ()
- void enable_interrupts ()
- void pic_mask (char enable)
- int com_open (int *e_flag, int baud_rate)
- int com_close (void)
- int com_read (char *buf_ptr, int *count_ptr)
- int com_write (char *buf_ptr, int *count_ptr)
- void serial_io ()
- void serial_write ()
- void serial_read ()

Variables

- dcb_t * DCB

4.52.1 Function Documentation

4.52.1.1 com_close()

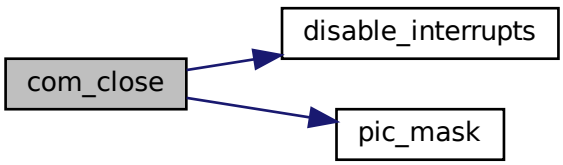
```
int com_close (
            void  )
```

+* com_close() Closes the communication port. +*

Returns

error code if port was not open, or a 0 for successful operation

Here is the call graph for this function:



4.52.1.2 com_open()

```
int com_open (
            int * e_flag,
            int baud_rate )
```

+∗ com_open() Opens the communication port. +∗

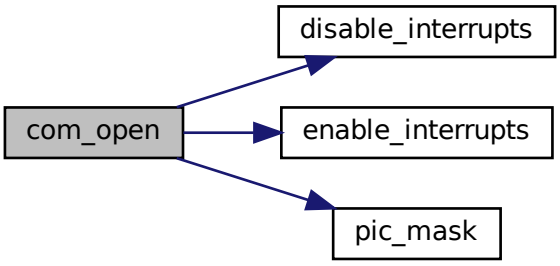
Parameters

e_flag	event flag will be set to 1 if read/write +∗
baud_rate	the desired baud rate +∗

Returns

Returns three possible error codes, or a 0 for successful operation.

Here is the call graph for this function:



4.52.1.3 com_read()

```
int com_read (
            char * buf_ptr,
            int * count_ptr )
```

+∗ com_read() Reads the buffer from the port. Non-blocking. +∗

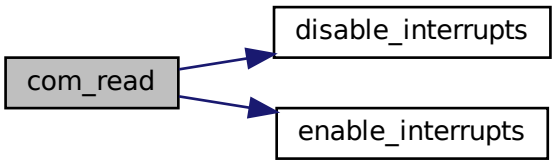
Parameters

buf_ptr	buffer in which the read characters will be stored. +∗
count_ptr	the maximum number of bytes to read. After completion, +∗ this will contain the number of characters read. +∗

Returns

Returns four possible error codes, or a 0 for successful operation.

Here is the call graph for this function:



4.52.1.4 com_write()

```
int com_write (
            char * buf_ptr,
            int * count_ptr )
```

+∗ com_write() Writes the buffer to the port. Non-blocking. +∗

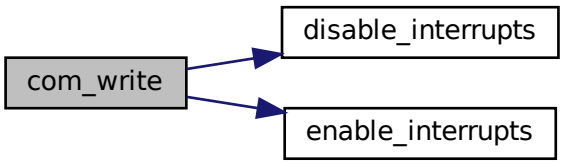
Parameters

buf_ptr	buffer in which the characters to write are stored. +∗
count_ptr	the number of characters from the buffer to write. +∗

Returns

Returns four possible error codes, or a 0 for successful operation.

Here is the call graph for this function:



4.52.1.5 disable_interrupts()

```
void disable_interrupts ( )
```

+∗ disable_interrupts() disables all interrupts to device.

4.52.1.6 enable_interrupts()

```
void enable_interrupts ( )
```

+∗ enable_interrupts() enables interrupts to device.

4.52.1.7 pic_mask()

```
void pic_mask (
            char enable )
```

+∗ pic_mask() masks so only the desired PIC interrupt is enabled or disabled. +∗

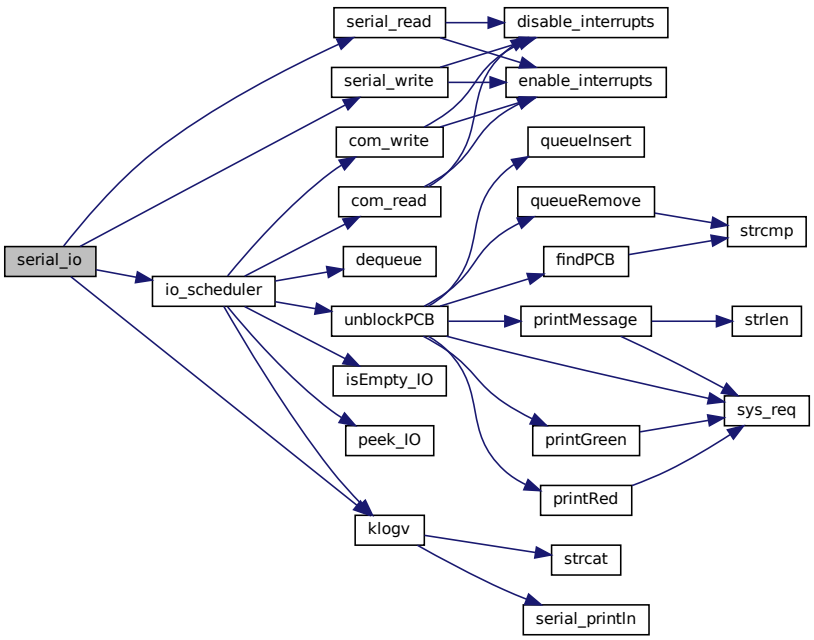
Parameters

enable	1 to enable or 0 to disable.

4.52.1.8 serial_io()

```
void serial_io ( )
```

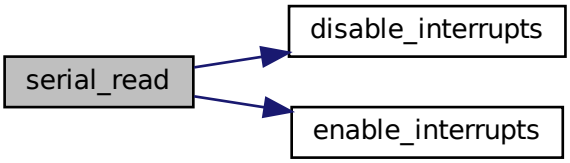
+∗ serial_io() is the interrupt C routine for serial IO. Here is the call graph for this function:



4.52.1.9 serial_read()

```
void serial_read ( )
```

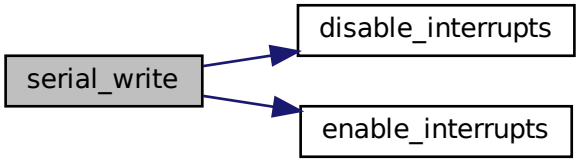
+∗ serial_read() provides interrupt routine for reading IO. Here is the call graph for this function:



4.52.1.10 serial_write()

```
void serial_write ( )
```

+∗ serial_write() provides interrupt routine for writing IO. Here is the call graph for this function:



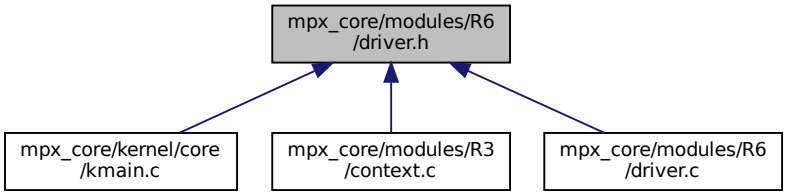
4.52.2 Variable Documentation

4.52.2.1 DCB

```
dcb_t* DCB
```

4.53 mpx_core/modules/R6/driver.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct dcb_s

Macros

- #define PIC_REG 0x20
- #define PIC_EOI 0x20
- #define PIC_MASK 0x21
- #define IRQ_COM1 0x10
- #define REG_RHR 0
- #define REG_THR 0
- #define REG_LSB 0
- #define REG_MSB 1
- #define REG_IER 1
- #define REG_IIR 2
- #define REG_LCR 3
- #define REG_MCR 4
- #define REG_LSR 5
- #define REG_MSR 6
- #define REG_SCRATCH 7

Typedefs

- typedef enum status_t status_t
- typedef struct dcb_s dcb_t

Enumerations

- enum status_t { STATUS_IDLE, STATUS_READING, STATUS_WRITING }

Functions

- void pic_mask (char enable)
- void disable_interrupts ()
- void enable_interrupts ()
- int com_open (int *e_flag, int baud_rate)
- int com_close (void)
- int com_read (char *buf_ptr, int *count_ptr)
- int com_write (char *buf_ptr, int *count_ptr)

- void serial_io ()
- void serial_write ()
- void serial_read ()

4.53.1 Macro Definition Documentation

4.53.1.1 IRQ_COM1

```
#define IRQ_COM1 0x10
```

4.53.1.2 PIC_EOI

```
#define PIC_EOI 0x20
```

4.53.1.3 PIC_MASK

```
#define PIC_MASK 0x21
```

4.53.1.4 PIC_REG

```
#define PIC_REG 0x20
```

4.53.1.5 REG_IER

```
#define REG_IER 1
```

4.53.1.6 REG_IIR

```
#define REG_IIR 2
```

4.53.1.7 REG_LCR

```
#define REG_LCR 3
```

4.53.1.8 REG_LSB

```
#define REG_LSB 0
```

4.53.1.9 REG_LSR

```
#define REG_LSR 5
```

4.53.1.10 REG_MCR

```
#define REG_MCR 4
```

4.53.1.11 REG_MSB

```
#define REG_MSB 1
```

4.53.1.12 REG_MSR

```
#define REG_MSR 6
```

4.53.1.13 REG_RHR

```
#define REG_RHR 0
```

4.53.1.14 REG_SCRATCH

```
#define REG_SCRATCH 7
```

4.53.1.15 REG_THR

```
#define REG_THR 0
```

4.53.2 Typedef Documentation

4.53.2.1 dcb_t

```
typedef struct dcb_s dcb_t
```

+∗ struct dcb represents a Device Control Block. +∗ A dcb should exist for each COM port, but you can just use COM1 +∗

Parameters

com_port	the COM port. (You can omit this and just always use COM1) +∗
port_open	whether the COM is open. +∗
e_flag	whether the operation has completed (0 or 1). +∗
status	the different operations (IDLE, READ, WRITE). +∗
buffer_ptr	the buffer array to read into/write from. +∗
count_ptr	how many characters to read/write. +∗
buffer_loc	the current location we are reading/writing at. +∗
byte_count	the number of bytes that have been read/written so far.

4.53.2.2 status_t

```
typedef enum status_t status_t
```

+∗ enum for the possible dcb states.

4.53.3 Enumeration Type Documentation

4.53.3.1 status_t

```
enum status_t
```

+∗ enum for the possible dcb states.

Enumerator

STATUS_IDLE	Port is idle
STATUS_READING	Port is reading
STATUS_WRITING	Port is writing

4.53.4 Function Documentation

4.53.4.1 com_close()

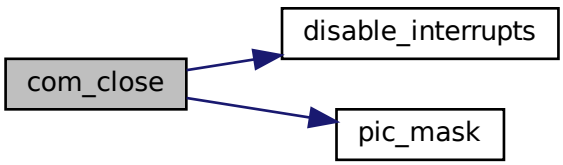
```
int com_close (
            void  )
```

+∗ com_close() Closes the communication port. +∗

Returns

error code if port was not open, or a 0 for successful operation

Here is the call graph for this function:



4.53.4.2 com_open()

```
int com_open (
            int * e_flag,
            int baud_rate )
```

+∗ com_open() Opens the communication port. +∗

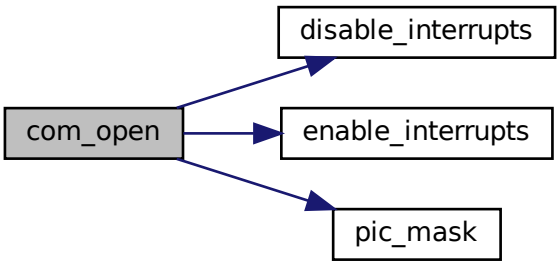
Parameters

e_flag	event flag will be set to 1 if read/write +∗
baud_rate	the desired baud rate +∗

Returns

Returns three possible error codes, or a 0 for successful operation.

Here is the call graph for this function:



4.53.4.3 com_read()

```
int com_read (
            char * buf_ptr,
            int * count_ptr )
```

+∗ com_read() Reads the buffer from the port. Non-blocking. +∗

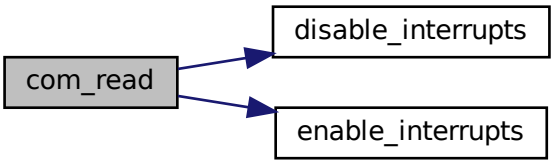
Parameters

buf_ptr	buffer in which the read characters will be stored. +∗
count_ptr	the maximum number of bytes to read. After completion, +∗ this will contain the number of characters read. +∗

Returns

Returns four possible error codes, or a 0 for successful operation.

Here is the call graph for this function:



4.53.4.4 com_write()

```
int com_write (
            char * buf_ptr,
            int * count_ptr )
```

+∗ com_write() Writes the buffer to the port. Non-blocking. +∗

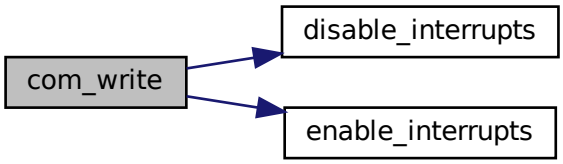
Parameters

buf_ptr	buffer in which the characters to write are stored. +∗
count_ptr	the number of characters from the buffer to write. +∗

Returns

Returns four possible error codes, or a 0 for successful operation.

Here is the call graph for this function:



4.53.4.5 disable_interrupts()

```
void disable_interrupts ( )
```

+∗ disable_interrupts() disables all interrupts to device.

4.53.4.6 enable_interrupts()

```
void enable_interrupts ( )
```

+∗ enable_interrupts() enables interrupts to device.

4.53.4.7 pic_mask()

```
void pic_mask (
            char enable )
```

+∗ pic_mask() masks so only the desired PIC interrupt is enabled or disabled. +∗

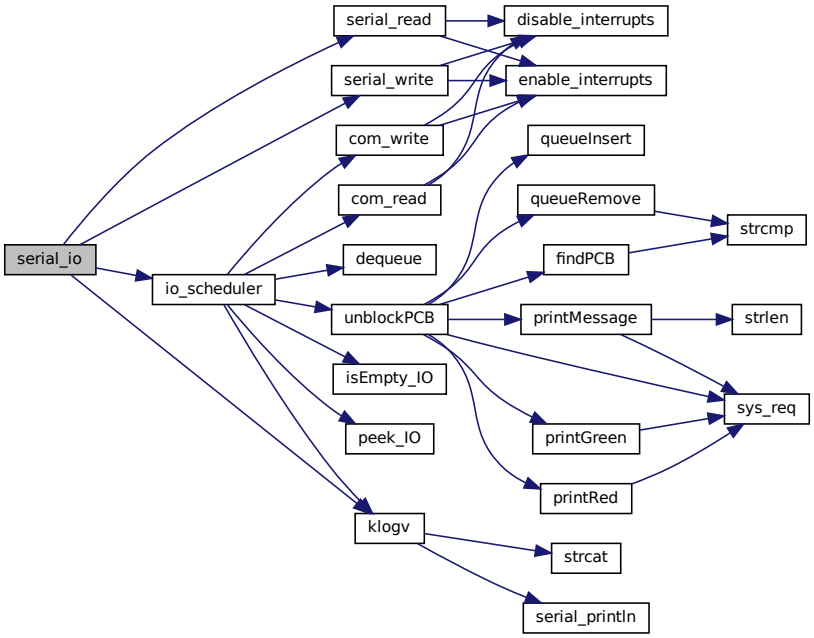
Parameters

enable	1 to enable or 0 to disable.

4.53.4.8 serial_io()

```
void serial_io ( )
```

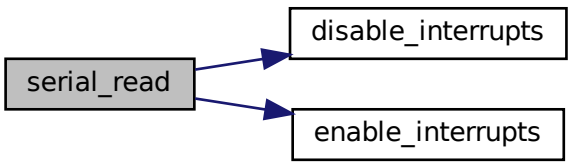
+∗ serial_io() is the interrupt C routine for serial IO. Here is the call graph for this function:



4.53.4.9 serial_read()

```
void serial_read ( )
```

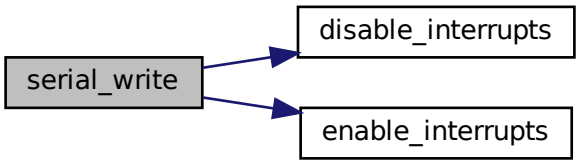
+∗ serial_read() provides interrupt routine for reading IO. Here is the call graph for this function:



4.53.4.10 serial_write()

```
void serial_write ( )
```

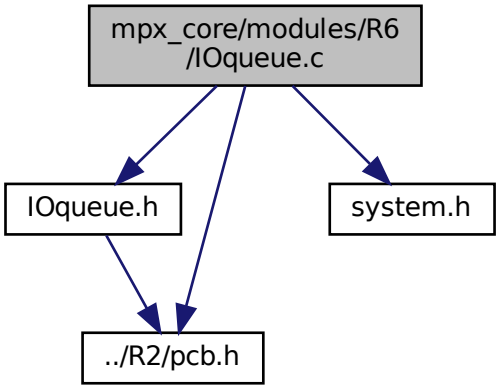
+∗ serial_write() provides interrupt routine for writing IO. Here is the call graph for this function:



4.54 mpx_core/modules/R6/IOqueue.c File Reference

```
#include "IOqueue.h"
#include <system.h>
#include "../R2/pcb.h"
```

Include dependency graph for IOqueue.c:



Functions

- void enqueue (ioqueue_t ∗queue, iod_t ∗item)
- iod_t ∗ dequeue (ioqueue_t ∗queue)
- iod_t ∗ peek_IO (ioqueue_t ∗queue)
- int isEmpty_IO (ioqueue_t ∗queue)

Variables

- ioqueue_t ioqueue ={NULL,NULL,0}

4.54.1 Function Documentation

4.54.1.1 dequeue()

```
iod_t* dequeue (
              ioqueue_t * queue )
```

4.54.1.2 enqueue()

```
void enqueue (
              ioqueue_t * queue,
              iod_t * item )
```

4.54.1.3 isEmpty_IO()

```
int isEmpty_IO (
              ioqueue_t * queue )
```

4.54.1.4 peek_IO()

```
iod_t* peek_IO (
              ioqueue_t * queue )
```

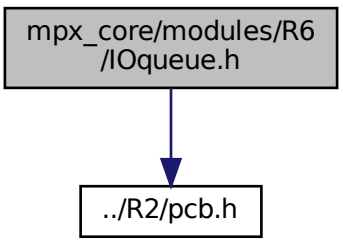
4.54.2 Variable Documentation

4.54.2.1 ioqueue

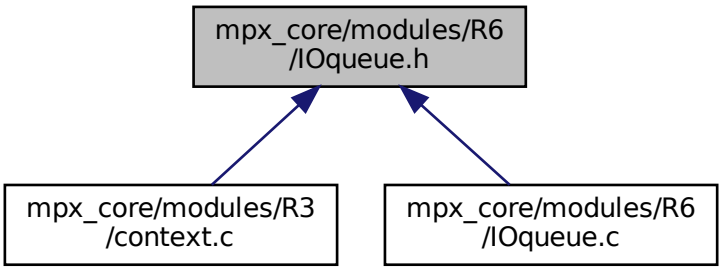
```
ioqueue_t ioqueue ={NULL,NULL,0}
```

4.55 mpx_core/modules/R6/IOqueue.h File Reference

```
#include "../R2/pcb.h"
```
Include dependency graph for IOqueue.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct iod_s
- struct ioqueue_s

Typedefs

- typedef struct iod_s iod_t
- typedef struct ioqueue_s ioqueue_t

Functions

- void enqueue (ioqueue_t *queue, iod_t *item)
- iod_t * dequeue (ioqueue_t *queue)
- iod_t * peek_IO (ioqueue_t *queue)
- int isEmpty_IO (ioqueue_t *queue)

4.55.1 Typedef Documentation

4.55.1.1 iod_t

```
typedef struct iod_s iod_t
```

4.55.1.2 ioqueue_t

```
typedef struct ioqueue_s ioqueue_t
```

4.55.2 Function Documentation

4.55.2.1 dequeue()

```
iod_t* dequeue (
            ioqueue_t * queue )
```

4.55.2.2 enqueue()

```
void enqueue (
            ioqueue_t * queue,
            iod_t * item )
```

4.55.2.3 isEmpty_IO()

```
int isEmpty_IO (
            ioqueue_t * queue )
```

4.55.2.4 peek_IO()

```
iod_t* peek_IO (
            ioqueue_t * queue )
```

Index

__attribute__
    tables.h, 53
__end
    heap.c, 106
_end
    heap.c, 106
_kmalloc
    heap.c, 105
    heap.h, 58

abs
    string.c, 112
access
    gdt_entry_struct, 16
    tables.h, 55
accessed
    page_entry, 29
addToHistory
    serial.c, 90
    serial.h, 42
alarm.c
    createAlarm, 227
    createAlarmCall, 228
    createAList, 229
    head_alarms, 231
    infiniteProcess, 229
    timeDaemon, 230
alarm.h
    alarm_t, 232
    createAlarm, 232
    createAlarmCall, 232
    createAList, 233
    infiniteProcess, 234
    timeDaemon, 234
alarm_s, 5
    message, 5
    next, 6
    prev, 6
    time, 6
alarm_t
    alarm.h, 232
aList, 6
    count, 7
    head, 7
    list.h, 238
    tail, 7
aListInsert
    list.c, 236
    list.h, 238
aListRemove
    list.c, 236
    list.h, 238
alloc
    heap.c, 105
    heap.h, 58
ALLOCATED_MEMORY
    memoryManagment.h, 259
allocateMemory
    memoryManagment.c, 253
    memoryManagment.h, 260
allocateMemoryCall
    memoryCommands.c, 248
    memoryCommands.h, 250
allocatePCB
    pcb.c, 165
    pcb.h, 170
APPLICATION
    pcb.h, 169
asm
    system.h, 68
atoi
    string.c, 112
    string.h, 63

BACKSPACE
    serial.c, 87
backspaceChar
    serial.c, 90
    serial.h, 42
base
    gdt_descriptor_struct, 16
    heap, 19
    idt_struct, 21
    tables.h, 55
base_high
    gdt_entry_struct, 17
    idt_entry_struct, 20
    tables.h, 55
base_low
    gdt_entry_struct, 17
    idt_entry_struct, 20
    tables.h, 55
base_mid
    gdt_entry_struct, 17
    tables.h, 55
bcdToDec
    time.c, 157
    time.h, 161
block
    index_entry, 21

BLOCKED
pcb.h, 169
blockedQueue
processManagement.c, 184
blockedTitle
queue.c, 199
blockPCB
processManagement.c, 174
processManagement.h, 185
blockPCBCall
R2Commands.c, 204
R2Commands.h, 209
border
queue.c, 199
borderSize
queue.c, 199
bounds
interrupts.c, 73
breakpoint
interrupts.c, 73
buffer_loc
dcb_s, 13
buffer_ptr
dcb_s, 14
iod_s, 24
param, 31
bufferIndex
serial.c, 99
bufferSize
serial.c, 99
byte_count
dcb_s, 14

cdir
paging.c, 110
class
PCB, 32
clear
miscCommands.c, 147
miscCommands.h, 152
CLEAR_ALL
miscCommands.c, 147
clear_bit
paging.c, 108
paging.h, 61
CLEAR_LINE
serial.c, 87
clearQueues
processManagement.c, 174
processManagement.h, 186
cli
system.h, 68
CMCB_s, 7
namePCB, 8
next, 8
prev, 8
size, 8
start, 8
type, 8
CMCB_t
memoryManagment.h, 260
cmdBuffer
comhand.c, 134
cmdBufferSize
comhand.c, 134
COM1
serial.h, 41
COM2
serial.h, 41
COM3
serial.h, 42
COM4
serial.h, 42
com_close
driver.c, 270
driver.h, 278
com_open
driver.c, 271
driver.h, 278
COM_PORT
mpx_supt.h, 122
com_port
dcb_s, 14
com_read
driver.c, 271
driver.h, 279
com_write
driver.c, 272
driver.h, 279
comEmpty
serial.c, 91
serial.h, 43
comhand
comhand.c, 129
comhand.h, 135
comhand.c
cmdBuffer, 134
cmdBufferSize, 134
comhand, 129
DOWN, 129
GREEN, 129
intro, 134
introSize, 134
noSuchCommand, 131
poll, 132
printGreen, 132
printRed, 133
printWhite, 133
RED, 129
resetBuffer, 133
UP, 129
WHITE, 129
comhand.h
comhand, 135
createPCBC, 137
deletePCBCall, 138
noSuchCommand, 138

poll, 138
printGreen, 139
printRed, 139
printWhite, 139
resetBuffer, 140
commands
serial.c, 99
Context, 9
context.h, 217
cs, 9
ds, 9
eax, 9
ebp, 10
ebx, 10
ecx, 10
edi, 10
edx, 10
eflags, 10
eip, 10
es, 10
esi, 11
esp, 11
fs, 11
gs, 11
context.c
cop, 216
DCB, 216
func_ptr, 213
io_scheduler, 213
ioqueue, 217
loadProcess, 214
loadr3, 215
old, 217
params, 217
sys_call, 215
yield, 216
context.h
Context, 217
io_scheduler, 217
loadr3, 218
sys_call, 219
yield, 220
cop
context.c, 216
coprocessor
interrupts.c, 73
coprocessor_segment
interrupts.c, 73
count
aList, 7
ioqueue_s, 25
mList, 27
Queue, 34
count_ptr
dcb_s, 14
iod_s, 24
param, 31
createAlarm
alarm.c, 227
alarm.h, 232
createAlarmCall
alarm.c, 228
alarm.h, 232
createAList
alarm.c, 229
alarm.h, 233
createmList
memoryManagment.c, 253
memoryManagment.h, 261
createPCB
processManagement.c, 175
processManagement.h, 186
createPCBC
comhand.h, 137
createPCBCall
R2Commands.c, 204
R2Commands.h, 209
createQueue
processManagement.c, 175
processManagement.h, 187
cs
Context, 9
curr_heap
heap.c, 106
current_module
mpx_supt.c, 120

date.c
getDate, 141
parseDate, 141
setDate, 142
setDateCall, 142
date.h
getDate, 144
parseDate, 144
setDate, 145
setDateCall, 145
date_time, 11
day_m, 12
day_w, 12
day_y, 12
hour, 12
min, 12
mon, 12
sec, 12
year, 12
day_m
date_time, 12
day_w
date_time, 12
day_y
date_time, 12
DCB
context.c, 216
driver.c, 274
dcb_s, 13
buffer_loc, 13

buffer_ptr, 14
byte_count, 14
com_port, 14
count_ptr, 14
e_flag, 14
port_open, 14
status, 14
dcb_t
driver.h, 277
debug
interrupts.c, 73
DEFAULT_DEVICE
mpx_supt.h, 122
DELETE
serial.c, 87
deleteChar
serial.c, 91
serial.h, 43
deletePCB
processManagement.c, 176
processManagement.h, 188
deletePCBCall
comhand.h, 138
R2Commands.c, 205
R2Commands.h, 209
dequeue
IOqueue.c, 282
IOqueue.h, 284
device_id
param, 31
device_not_available
interrupts.c, 73
dirty
page_entry, 29
disable_interrupts
driver.c, 272
driver.h, 280
divide_error
interrupts.c, 74
do_bounds
interrupts.c, 74
do_breakpoint
interrupts.c, 74
do_coprocessor
interrupts.c, 74
do_coprocessor_segment
interrupts.c, 75
do_debug
interrupts.c, 75
do_device_not_available
interrupts.c, 75
do_divide_error
interrupts.c, 76
do_double_fault
interrupts.c, 76
do_general_protection
interrupts.c, 76
do_invalid_op
interrupts.c, 77
do_invalid_tss
interrupts.c, 77
do_isr
interrupts.c, 77
do_nmi
interrupts.c, 78
do_overflow
interrupts.c, 78
do_page_fault
interrupts.c, 79
do_reserved
interrupts.c, 79
do_segment_not_present
interrupts.c, 79
do_stack_segment
interrupts.c, 79
double_fault
interrupts.c, 80
DOWN
comhand.c, 129
serial.c, 87
DOWN_ARROW
serial.c, 88
driver.c
com_close, 270
com_open, 271
com_read, 271
com_write, 272
DCB, 274
disable_interrupts, 272
enable_interrupts, 272
pic_mask, 273
serial_io, 273
serial_read, 273
serial_write, 274
driver.h
com_close, 278
com_open, 278
com_read, 279
com_write, 279
dcb_t, 277
disable_interrupts, 280
enable_interrupts, 280
IRQ_COM1, 276
PIC_EOI, 276
PIC_MASK, 276
pic_mask, 280
PIC_REG, 276
REG_IER, 276
REG_IIR, 276
REG_LCR, 276
REG_LSB, 276
REG_LSR, 276
REG_MCR, 276
REG_MSB, 276
REG_MSR, 277
REG_RHR, 277

REG_SCRATCH, 277
REG_THR, 277
serial_io, 280
serial_read, 281
serial_write, 281
STATUS_IDLE, 278
STATUS_READING, 278
status_t, 277
STATUS_WRITING, 278
ds
Context, 9

e_flag
dcb_s, 14
eax
Context, 9
ebp
Context, 10
ebx
Context, 10
ecx
Context, 10
edi
Context, 10
edx
Context, 10
eflags
Context, 10
eip
Context, 10
empty
index_entry, 22
enable_interrupts
driver.c, 272
driver.h, 280
end
heap.c, 107
enqueue
IOqueue.c, 283
IOqueue.h, 284
ENTER
serial.c, 88
er1
test.c, 223
er2
test.c, 223
er3
test.c, 223
er4
test.c, 223
er5
test.c, 223
erSize
test.c, 223
es
Context, 10
ESCAPE
serial.c, 88
esi
Context, 11
esp
Context, 11
EXIT
mpx_supt.h, 122

FALSE
mpx_supt.h, 122
find_free
paging.c, 108
findPCB
queue.c, 196
queue.h, 200
first_free
paging.h, 61
flags
gdt_entry_struct, 17
idt_entry_struct, 20
tables.h, 56
footer, 15
head, 15
frameaddr
page_entry, 29
frames
paging.c, 110
FREE_MEMORY
memoryManagment.h, 259
freeMem
memoryManagment.c, 254
memoryManagment.h, 261
FreeMemoryCall
memoryCommands.c, 248
memoryCommands.h, 251
freePCB
pcb.c, 165
pcb.h, 170
fs
Context, 11
func_ptr
context.c, 213

GDT_CS_ID
system.h, 68
gdt_descriptor_struct, 16
base, 16
limit, 16
GDT_DS_ID
system.h, 68
gdt_entries
tables.c, 103
gdt_entry_struct, 16
access, 16
base_high, 17
base_low, 17
base_mid, 17
flags, 17
limit_low, 17
gdt_init_entry
tables.c, 102

tables.h, 53
gdt_ptr
tables.c, 103
general_protection
interrupts.c, 80
get_bit
paging.c, 108
paging.h, 61
get_page
paging.c, 108
paging.h, 61
getCommand
serial.h, 43
getCommandDown
serial.c, 91
serial.h, 44
getCommandUp
serial.c, 92
serial.h, 44
getDate
date.c, 141
date.h, 144
getHistory
serial.c, 92
serial.h, 44
getNextReady
processManagement.c, 177
processManagement.h, 188
getTime
time.c, 157
time.h, 161
getTimeCall
miscCommands.c, 147
miscCommands.h, 152
GREEN
comhand.c, 129
gs
Context, 11

head
aList, 7
footer, 15
ioqueue_s, 25
mList, 27
Queue, 34
head_alarms
alarm.c, 231
header, 17
index_id, 18
size, 18
heap, 18
base, 19
index, 19
max_size, 19
min_size, 19
heap.c
__end, 106
_end, 106
_kmalloc, 105
alloc, 105
curr_heap, 106
end, 107
kdir, 107
kheap, 107
kmalloc, 105
make_heap, 106
phys_alloc_addr, 107
heap.h
_kmalloc, 58
alloc, 58
init_kheap, 58
kfree, 59
KHEAP_BASE, 57
KHEAP_MIN, 57
KHEAP_SIZE, 57
kmalloc, 59
make_heap, 59
TABLE_SIZE, 57
HEAP_SIZE
memoryManagment.h, 259
heap_top
memoryManagment.c, 258
heapIsEmpty
memoryManagment.c, 254
memoryManagment.h, 262
help
miscCommands.c, 148
miscCommands.h, 153
help2
miscCommands.c, 149
miscCommands.h, 154
help3
miscCommands.c, 149
miscCommands.h, 154
help4
miscCommands.c, 149
miscCommands.h, 154
help5
miscCommands.c, 149
miscCommands.h, 154
historyFull
serial.c, 92
serial.h, 44
hlt
system.h, 68
hour
date_time, 12

ICW1
interrupts.c, 72
ICW4
interrupts.c, 72
id
index_table, 23
IDLE
mpx_supt.h, 122
idle
mpx_supt.c, 117

mpx_supt.h, 124
idt_entries
interrupts.c, 83
tables.c, 103
idt_entry_struct, 19
base_high, 20
base_low, 20
flags, 20
sselect, 20
zero, 20
idt_ptr
tables.c, 104
idt_set_gate
tables.c, 102
tables.h, 54
idt_struct, 21
base, 21
limit, 21
inb
io.h, 40
index
heap, 19
serial.c, 99
index_entry, 21
block, 21
empty, 22
size, 22
index_id
header, 18
index_table, 22
id, 23
table, 23
infiniteProcess
alarm.c, 229
alarm.h, 234
init_gdt
tables.c, 102
tables.h, 54
init_idt
tables.c, 102
tables.h, 54
init_irq
interrupts.c, 80
interrupts.h, 38
init_kheap
heap.h, 58
init_paging
paging.c, 109
paging.h, 61
init_pic
interrupts.c, 81
interrupts.h, 39
init_serial
serial.c, 93
serial.h, 45
initializeHeap
memoryManagment.c, 255
memoryManagment.h, 262
initializeHistory
serial.c, 93
serial.h, 45
initQueues
processManagement.c, 177
processManagement.h, 189
insertPCB
processManagement.c, 178
processManagement.h, 189
interrupts.c
bounds, 73
breakpoint, 73
coprocessor, 73
coprocessor_segment, 73
debug, 73
device_not_available, 73
divide_error, 74
do_bounds, 74
do_breakpoint, 74
do_coprocessor, 74
do_coprocessor_segment, 75
do_debug, 75
do_device_not_available, 75
do_divide_error, 76
do_double_fault, 76
do_general_protection, 76
do_invalid_op, 77
do_invalid_tss, 77
do_isr, 77
do_nmi, 78
do_overflow, 78
do_page_fault, 79
do_reserved, 79
do_segment_not_present, 79
do_stack_segment, 79
double_fault, 80
general_protection, 80
ICW1, 72
ICW4, 72
idt_entries, 83
init_irq, 80
init_pic, 81
invalid_op, 81
invalid_tss, 82
io_wait, 72
isr0, 82
nmi, 82
overflow, 82
page_fault, 82
PIC1, 72
PIC2, 73
reserved, 82
rtc_isr, 82
segment_not_present, 82
serial_io_isr, 83
stack_segment, 83
sys_call_isr, 83
interrupts.h

init_irq, 38
init_pic, 39
intro
comhand.c, 134
introSize
comhand.c, 134
INVALID_BUFFER
mpx_supt.h, 122
INVALID_COUNT
mpx_supt.h, 123
invalid_op
interrupts.c, 81
INVALID_OPERATION
mpx_supt.h, 123
invalid_tss
interrupts.c, 82
io.h
inb, 40
outb, 40
IO_MODULE
mpx_supt.h, 123
io_scheduler
context.c, 213
context.h, 217
io_wait
interrupts.c, 72
iod_s, 23
buffer_ptr, 24
count_ptr, 24
name, 24
next, 24
op_code, 24
pcb, 24
iod_t
IOqueue.h, 284
ioqueue
context.c, 217
IOqueue.c, 283
IOqueue.c
dequeue, 282
enqueue, 283
ioqueue, 283
isEmpty_IO, 283
peek_IO, 283
IOqueue.h
dequeue, 284
enqueue, 284
iod_t, 284
ioqueue_t, 284
isEmpty_IO, 285
peek_IO, 285
ioqueue_s, 25
count, 25
head, 25
tail, 25
ioqueue_t
IOqueue.h, 284
iret
system.h, 68
IRQ_COM1
driver.h, 276
is_io_module_active
mpx_supt.c, 117
mpx_supt.h, 125
isBackspace
serial.c, 93
serial.h, 45
isDownArrow
serial.c, 93
serial.h, 45
isEmpty
queue.c, 197
queue.h, 201
isEmpty_IO
IOqueue.c, 283
IOqueue.h, 285
isEmptyCall
memoryCommands.c, 249
memoryCommands.h, 251
isEnter
serial.c, 93
serial.h, 45
isEscape
serial.c, 93
serial.h, 45
isLeftArrow
serial.c, 93
serial.h, 45
isLetter
serial.c, 94
serial.h, 46
isr0
interrupts.c, 82
isRightArrow
serial.c, 94
serial.h, 46
isspace
string.c, 112
string.h, 64
isUpArrow
serial.c, 94
serial.h, 46
itoa
string.c, 113
string.h, 64

kdir
heap.c, 107
paging.c, 110
kfree
heap.h, 59
kheap
heap.c, 107
paging.c, 110
KHEAP_BASE
heap.h, 57
KHEAP_MIN

heap.h, 57
KHEAP_SIZE
heap.h, 57
klogv
system.c, 100
system.h, 70
kmain
kmain.c, 84
kmain.c
kmain, 84
kmalloc
heap.c, 105
heap.h, 59
kpanic
system.c, 101
system.h, 70

LEFT_ARROW
serial.c, 88
LEFT_BRACKET
serial.c, 88
leftArrowChar
serial.c, 94
serial.h, 46
letterChar
serial.c, 95
serial.h, 47
limit
gdt_descriptor_struct, 16
idt_struct, 21
tables.h, 56
limit_low
gdt_entry_struct, 17
tables.h, 56
list.c
aListInsert, 236
aListRemove, 236
list.h
aList, 238
aListInsert, 238
aListRemove, 238
LMCB_s, 26
size, 26
type, 26
LMCB_t
memoryManagment.h, 260
load_page_dir
paging.c, 109
paging.h, 62
loadComhand
loadComhand.c, 239
loadComhand.h, 243
loadComhand.c
loadComhand, 239
loadIdle, 240
loadInfiniteProcess, 241
loadTimeDaemon, 242
loadComhand.h
loadComhand, 243
loadIdle, 245
loadInfiniteProcess, 245
loadTimeDaemon, 246
loadIdle
loadComhand.c, 240
loadComhand.h, 245
loadInfiniteProcess
loadComhand.c, 241
loadComhand.h, 245
loadProcess
context.c, 214
loadr3
context.c, 215
context.h, 218
loadTimeDaemon
loadComhand.c, 242
loadComhand.h, 246

make_heap
heap.c, 106
heap.h, 59
max
serial.c, 88
MAX_SIZE
serial.c, 88
max_size
heap, 19
MCBList
memoryManagment.c, 258
MEM_MODULE
mpx_supt.h, 123
mem_size
paging.c, 111
memoryCommands.c
allocateMemoryCall, 248
FreeMemoryCall, 248
isEmptyCall, 249
memoryCommands.h
allocateMemoryCall, 250
FreeMemoryCall, 251
isEmptyCall, 251
memoryManagment.c
allocateMemory, 253
createmList, 253
freeMem, 254
heap_top, 258
heapIsEmpty, 254
initializeHeap, 255
MCBList, 258
merge, 255
printAll, 256
printAllocated, 256
printCMCB, 257
printFree, 257
memoryManagment.h
ALLOCATED_MEMORY, 259
allocateMemory, 260
CMCB_t, 260
createmList, 261

FREE_MEMORY, 259
freeMem, 261
HEAP_SIZE, 259
heapIsEmpty, 262
initializeHeap, 262
LMCB_t, 260
merge, 263
printAll, 264
printAllocated, 264
printCMCB, 265
printFree, 265
memset
string.c, 113
string.h, 65
merge
memoryManagment.c, 255
memoryManagment.h, 263
message
alarm_s, 5
min
date_time, 12
min_size
heap, 19
miscCommands.c
clear, 147
CLEAR_ALL, 147
getTimeCall, 147
help, 148
help2, 149
help3, 149
help4, 149
help5, 149
MOVE_DEFAULT, 147
showPCBCall, 150
shutdown, 150
version, 151
miscCommands.h
clear, 152
getTimeCall, 152
help, 153
help2, 154
help3, 154
help4, 154
help5, 154
showPCBCall, 155
shutdown, 155
version, 156
mList, 27
count, 27
head, 27
mList.h, 268
tail, 27
mList.c
mListInsert, 266
mListRemove, 267
mList.h
mList, 268
mListInsert, 269
mListRemove, 269
mListInsert
mList.c, 266
mList.h, 269
mListRemove
mList.c, 267
mList.h, 269
MODULE_F
mpx_supt.h, 123
MODULE_R1
mpx_supt.h, 123
MODULE_R2
mpx_supt.h, 123
MODULE_R3
mpx_supt.h, 123
MODULE_R4
mpx_supt.h, 124
MODULE_R5
mpx_supt.h, 124
mon
date_time, 12
MOVE_DEFAULT
miscCommands.c, 147
mpx_core/include/core/asm.h, 37
mpx_core/include/core/interrupts.h, 38
mpx_core/include/core/io.h, 40
mpx_core/include/core/serial.h, 40
mpx_core/include/core/tables.h, 52
mpx_core/include/mem/heap.h, 56
mpx_core/include/mem/paging.h, 60
mpx_core/include/string.h, 63
mpx_core/include/system.h, 67
mpx_core/kernel/core/interrupts.c, 71
mpx_core/kernel/core/kmain.c, 83
mpx_core/kernel/core/serial.c, 85
mpx_core/kernel/core/system.c, 100
mpx_core/kernel/core/tables.c, 101
mpx_core/kernel/mem/heap.c, 104
mpx_core/kernel/mem/paging.c, 107
mpx_core/lib/string.c, 111
mpx_core/modules/mpx_supt.c, 116
mpx_core/modules/mpx_supt.h, 121
mpx_core/modules/R1/comhand.c, 128
mpx_core/modules/R1/comhand.h, 135
mpx_core/modules/R1/date.c, 140
mpx_core/modules/R1/date.h, 143
mpx_core/modules/R1/miscCommands.c, 146
mpx_core/modules/R1/miscCommands.h, 152
mpx_core/modules/R1/time.c, 157
mpx_core/modules/R1/time.h, 160
mpx_core/modules/R2/pcb.c, 164
mpx_core/modules/R2/pcb.h, 168
mpx_core/modules/R2/processManagement.c, 173
mpx_core/modules/R2/processManagement.h, 184
mpx_core/modules/R2/queue.c, 195
mpx_core/modules/R2/queue.h, 200
mpx_core/modules/R2/R2Commands.c, 203
mpx_core/modules/R2/R2Commands.h, 208

mpx_core/modules/R3/context.c, 212
mpx_core/modules/R3/context.h, 217
mpx_core/modules/R3/test.c, 220
mpx_core/modules/R3/test.h, 224
mpx_core/modules/R4/alarm.c, 226
mpx_core/modules/R4/alarm.h, 231
mpx_core/modules/R4/list.c, 235
mpx_core/modules/R4/list.h, 237
mpx_core/modules/R4/loadComhand.c, 238
mpx_core/modules/R4/loadComhand.h, 243
mpx_core/modules/R5/memoryCommands.c, 247
mpx_core/modules/R5/memoryCommands.h, 250
mpx_core/modules/R5/memoryManagment.c, 252
mpx_core/modules/R5/memoryManagment.h, 258
mpx_core/modules/R5/mList.c, 266
mpx_core/modules/R5/mList.h, 267
mpx_core/modules/R6/driver.c, 269
mpx_core/modules/R6/driver.h, 275
mpx_core/modules/R6/IOqueue.c, 282
mpx_core/modules/R6/IOqueue.h, 283
mpx_init
  mpx_supt.c, 117
  mpx_supt.h, 125
mpx_supt.c
  current_module, 120
  idle, 117
  is_io_module_active, 117
  mpx_init, 117
  params, 120
  printlnMessage, 117
  printMessage, 118
  student_free, 120
  student_malloc, 120
  sys_alloc_mem, 119
  sys_free_mem, 119
  sys_req, 119
  sys_set_free, 119
  sys_set_malloc, 119
mpx_supt.h
  COM_PORT, 122
  DEFAULT_DEVICE, 122
  EXIT, 122
  FALSE, 122
  IDLE, 122
  idle, 124
  INVALID_BUFFER, 122
  INVALID_COUNT, 123
  INVALID_OPERATION, 123
  IO_MODULE, 123
  is_io_module_active, 125
  MEM_MODULE, 123
  MODULE_F, 123
  MODULE_R1, 123
  MODULE_R2, 123
  MODULE_R3, 123
  MODULE_R4, 124
  MODULE_R5, 124
  mpx_init, 125
  NO_ERROR, 124
  printlnMessage, 125
  printMessage, 126
  READ, 124
  sys_alloc_mem, 127
  sys_free_mem, 127
  sys_req, 127
  sys_set_free, 127
  sys_set_malloc, 127
  TRUE, 124
  WRITE, 124
msg1
  test.c, 224
msg2
  test.c, 224
msg3
  test.c, 224
msg4
  test.c, 224
msg5
  test.c, 224
msgSize
  test.c, 224
MV_CURSOR_LEFT
  serial.c, 88
MV_CURSOR_RIGHT
  serial.c, 89
MV_CURSOR_START
  serial.c, 89

name
  iod_s, 24
  PCB, 32
  Queue, 34
namePCB
  CMCB_s, 8
new_frame
  paging.c, 109
  paging.h, 62
NEWLINE
  serial.c, 89
next
  alarm_s, 6
  CMCB_s, 8
  iod_s, 24
  PCB, 32
nframes
  paging.c, 111
nmi
  interrupts.c, 82
NO_ERROR
  mpx_supt.h, 124
  serial.c, 89
no_warn
  system.h, 68
nop
  system.h, 68
noQueueError
  processManagement.c, 178

processManagement.h, 190
noSuchCommand
comhand.c, 131
comhand.h, 138
NOTSUSPENDED
pcb.h, 169
NULL
system.h, 69

old
context.c, 217
op_code
iod_s, 24
param, 31
outb
io.h, 40
overflow
interrupts.c, 82

page_dir, 28
tables, 28
tables_phys, 28
page_entry, 29
accessed, 29
dirty, 29
frameaddr, 29
present, 29
reserved, 29
usermode, 29
writeable, 30
page_fault
interrupts.c, 82
PAGE_SIZE
paging.h, 61
page_size
paging.c, 111
page_table, 30
pages, 30
pages
page_table, 30
paging.c
cdir, 110
clear_bit, 108
find_free, 108
frames, 110
get_bit, 108
get_page, 108
init_paging, 109
kdir, 110
kheap, 110
load_page_dir, 109
mem_size, 111
new_frame, 109
nframes, 111
page_size, 111
phys_alloc_addr, 111
set_bit, 110
paging.h
clear_bit, 61
first_free, 61
get_bit, 61
get_page, 61
init_paging, 61
load_page_dir, 62
new_frame, 62
PAGE_SIZE, 61
set_bit, 62
param, 31
buffer_ptr, 31
count_ptr, 31
device_id, 31
op_code, 31
params
context.c, 217
mpx_supt.c, 120
parseDate
date.c, 141
date.h, 144
parseTime
time.c, 158
time.h, 161
PCB, 32
class, 32
name, 32
next, 32
pcb.h, 170
prev, 32
priority, 33
stack, 33
stackBase, 33
stackTop, 33
status, 33
suspended, 33
pcb
iod_s, 24
pcb.c
allocatePCB, 165
freePCB, 165
printPCB, 166
setupPCB, 167
pcb.h
allocatePCB, 170
APPLICATION, 169
BLOCKED, 169
freePCB, 170
NOTSUSPENDED, 169
PCB, 170
printPCB, 171
READY, 169
RUNNING, 169
setupPCB, 172
stackSize, 169
SUSPENDED, 169
SYSTEM, 169
peek
queue.c, 197
queue.h, 201

peek_IO
IOqueue.c, 283
IOqueue.h, 285
phys_alloc_addr
heap.c, 107
paging.c, 111
PIC1
interrupts.c, 72
PIC2
interrupts.c, 73
PIC_EOI
driver.h, 276
PIC_MASK
driver.h, 276
pic_mask
driver.c, 273
driver.h, 280
PIC_REG
driver.h, 276
poll
comhand.c, 132
comhand.h, 138
polling
serial.c, 95
serial.h, 47
port_open
dcb_s, 14
present
page_entry, 29
prev
alarm_s, 6
CMCB_s, 8
PCB, 32
PRINT_GREEN
serial.c, 89
printAll
memoryManagment.c, 256
memoryManagment.h, 264
printAllocated
memoryManagment.c, 256
memoryManagment.h, 264
printCMCB
memoryManagment.c, 257
memoryManagment.h, 265
printFree
memoryManagment.c, 257
memoryManagment.h, 265
printGreen
comhand.c, 132
comhand.h, 139
printInMessage
mpx_supt.c, 117
mpx_supt.h, 125
printMessage
mpx_supt.c, 118
mpx_supt.h, 126
printPCB
pcb.c, 166
pcb.h, 171
printQueue
queue.c, 197
queue.h, 201
printRed
comhand.c, 133
comhand.h, 139
printWhite
comhand.c, 133
comhand.h, 139
priority
PCB, 33
proc1
test.c, 222
test.h, 225
proc2
test.c, 222
test.h, 225
proc3
test.c, 222
test.h, 225
proc4
test.c, 222
test.h, 225
proc5
test.c, 223
test.h, 226
processManagement.c
blockedQueue, 184
blockPCB, 174
clearQueues, 174
createPCB, 175
createQueue, 175
deletePCB, 176
getNextReady, 177
initQueues, 177
insertPCB, 178
noQueueError, 178
queuesCleared, 184
readyQueue, 184
removeBlocked, 179
removeReady, 179
resumePCB, 179
setPCBpriority, 180
showAll, 181
showBlocked, 181
showPCB, 181
showReady, 182
suspendPCB, 183
unblockPCB, 183
processManagement.h
blockPCB, 185
clearQueues, 186
createPCB, 186
createQueue, 187
deletePCB, 188
getNextReady, 188
initQueues, 189

insertPCB, 189
noQueueError, 190
removeBlocked, 190
removeReady, 190
resumePCB, 191
setPCBpriority, 191
showAll, 192
showBlocked, 192
showPCB, 193
showReady, 194
suspendPCB, 194
unblockPCB, 195

Queue, 34
count, 34
head, 34
name, 34
queue.h, 200
tail, 34
queue.c
blockedTitle, 199
border, 199
borderSize, 199
findPCB, 196
isEmpty, 197
peek, 197
printQueue, 197
queueInsert, 198
queueRemove, 199
readyTitle, 199
titleSize, 199
queue.h
findPCB, 200
isEmpty, 201
peek, 201
printQueue, 201
Queue, 200
queueInsert, 202
queueRemove, 203
queueInsert
queue.c, 198
queue.h, 202
queueRemove
queue.c, 199
queue.h, 203
queuesCleared
processManagement.c, 184

R2Commands.c
blockPCBCall, 204
createPCBCall, 204
deletePCBCall, 205
resumePCBCall, 206
setPriorityCall, 206
showReadyCall, 206
suspendPCBCall, 207
unblockPCBCall, 207
R2Commands.h
blockPCBCall, 209
createPCBCall, 209
deletePCBCall, 209
resumePCBCall, 210
setPriorityCall, 211
showReadyCall, 211
suspendPCBCall, 211
unblockPCBCall, 212
RC_1
test.c, 221
RC_2
test.c, 221
RC_3
test.c, 221
RC_4
test.c, 221
RC_5
test.c, 221
READ
mpx_supt.h, 124
READY
pcb.h, 169
readyQueue
processManagement.c, 184
readyTitle
queue.c, 199
RED
comhand.c, 129
REG_IER
driver.h, 276
REG_IIR
driver.h, 276
REG_LCR
driver.h, 276
REG_LSB
driver.h, 276
REG_LSR
driver.h, 276
REG_MCR
driver.h, 276
REG_MSB
driver.h, 276
REG_MSR
driver.h, 277
REG_RHR
driver.h, 277
REG_SCRATCH
driver.h, 277
REG_THR
driver.h, 277
removeBlocked
processManagement.c, 179
processManagement.h, 190
removeReady
processManagement.c, 179
processManagement.h, 190
reserved
interrupts.c, 82
page_entry, 29

resetBuffer
comhand.c, 133
comhand.h, 140
resetCursor
serial.c, 96
serial.h, 49
resetLine
serial.c, 97
serial.h, 50
resumePCB
processManagement.c, 179
processManagement.h, 191
resumePCBCall
R2Commands.c, 206
R2Commands.h, 210
reverse
string.c, 114
RIGHT_ARROW
serial.c, 89
rightArrowChar
serial.c, 97
serial.h, 51
rtc_isr
interrupts.c, 82
RUNNING
pcb.h, 169

sec
date_time, 12
segment_not_present
interrupts.c, 82
serial.c
addToHistory, 90
BACKSPACE, 87
backspaceChar, 90
bufferIndex, 99
bufferSize, 99
CLEAR_LINE, 87
comEmpty, 91
commands, 99
DELETE, 87
deleteChar, 91
DOWN, 87
DOWN_ARROW, 88
ENTER, 88
ESCAPE, 88
getCommandDown, 91
getCommandUp, 92
getHistory, 92
historyFull, 92
index, 99
init_serial, 93
initializeHistory, 93
isBackspace, 93
isDownArrow, 93
isEnter, 93
isEscape, 93
isLeftArrow, 93
isLetter, 94
isRightArrow, 94
isUpArrow, 94
LEFT_ARROW, 88
LEFT_BRACKET, 88
leftArrowChar, 94
letterChar, 95
max, 88
MAX_SIZE, 88
MV_CURSOR_LEFT, 88
MV_CURSOR_RIGHT, 89
MV_CURSOR_START, 89
NEWLINE, 89
NO_ERROR, 89
polling, 95
PRINT_GREEN, 89
resetCursor, 96
resetLine, 97
RIGHT_ARROW, 89
rightArrowChar, 97
serial_port_in, 99
serial_port_out, 99
serial_print, 98
serial_println, 98
set_serial_in, 98
set_serial_out, 98
size, 99
UP, 89
UP_ARROW, 89
serial.h
addToHistory, 42
backspaceChar, 42
COM1, 41
COM2, 41
COM3, 42
COM4, 42
comEmpty, 43
deleteChar, 43
getCommand, 43
getCommandDown, 44
getCommandUp, 44
getHistory, 44
historyFull, 44
init_serial, 45
initializeHistory, 45
isBackspace, 45
isDownArrow, 45
isEnter, 45
isEscape, 45
isLeftArrow, 45
isLetter, 46
isRightArrow, 46
isUpArrow, 46
leftArrowChar, 46
letterChar, 47
polling, 47
resetCursor, 49
resetLine, 50
rightArrowChar, 51

serial_print, 51
serial_println, 52
set_serial_in, 52
set_serial_out, 52
serial_io
driver.c, 273
driver.h, 280
serial_io_isr
interrupts.c, 83
serial_port_in
serial.c, 99
serial_port_out
serial.c, 99
serial_print
serial.c, 98
serial.h, 51
serial_println
serial.c, 98
serial.h, 52
serial_read
driver.c, 273
driver.h, 281
serial_write
driver.c, 274
driver.h, 281
set_bit
paging.c, 110
paging.h, 62
set_serial_in
serial.c, 98
serial.h, 52
set_serial_out
serial.c, 98
serial.h, 52
setDate
date.c, 142
date.h, 145
setDateCall
date.c, 142
date.h, 145
setPCBpriority
processManagement.c, 180
processManagement.h, 191
setPriorityCall
R2Commands.c, 206
R2Commands.h, 211
setTime
time.c, 159
time.h, 163
setTimeCall
time.c, 159
time.h, 163
setupPCB
pcb.c, 167
pcb.h, 172
showAll
processManagement.c, 181
processManagement.h, 192
showBlocked
processManagement.c, 181
processManagement.h, 192
showPCB
processManagement.c, 181
processManagement.h, 193
showPCBCall
miscCommands.c, 150
miscCommands.h, 155
showReady
processManagement.c, 182
processManagement.h, 194
showReadyCall
R2Commands.c, 206
R2Commands.h, 211
shutdown
miscCommands.c, 150
miscCommands.h, 155
size
CMCB_s, 8
header, 18
index_entry, 22
LMCB_s, 26
serial.c, 99
size_t
system.h, 69
sselect
idt_entry_struct, 20
tables.h, 56
stack
PCB, 33
stack_segment
interrupts.c, 83
stackBase
PCB, 33
stackSize
pcb.h, 169
stackTop
PCB, 33
start
CMCB_s, 8
status
dcb_s, 14
PCB, 33
STATUS_IDLE
driver.h, 278
STATUS_READING
driver.h, 278
status_t
driver.h, 277
STATUS_WRITING
driver.h, 278
sti
system.h, 69
strcasecmp
string.c, 114
string.h, 65
strcat

string.c, 114
string.h, 65
strcmp
string.c, 115
string.h, 66
strcpy
string.c, 115
string.h, 66
string.c
abs, 112
atoi, 112
isspace, 112
itoa, 113
memset, 113
reverse, 114
strcasecmp, 114
strcat, 114
strcmp, 115
strcpy, 115
strlen, 115
strtok, 115
swap, 115
toupper, 115
string.h
atoi, 63
isspace, 64
itoa, 64
memset, 65
strcasecmp, 65
strcat, 65
strcmp, 66
strcpy, 66
strlen, 66
strtok, 66
toupper, 66
strlen
string.c, 115
string.h, 66
strtok
string.c, 115
string.h, 66
student_free
mpx_supt.c, 120
student_malloc
mpx_supt.c, 120
SUSPENDED
pcb.h, 169
suspended
PCB, 33
suspendPCB
processManagement.c, 183
processManagement.h, 194
suspendPCBCall
R2Commands.c, 207
R2Commands.h, 211
swap
string.c, 115
sys_alloc_mem
mpx_supt.c, 119
mpx_supt.h, 127
sys_call
context.c, 215
context.h, 219
sys_call_isr
interrupts.c, 83
sys_free_mem
mpx_supt.c, 119
mpx_supt.h, 127
sys_req
mpx_supt.c, 119
mpx_supt.h, 127
sys_set_free
mpx_supt.c, 119
mpx_supt.h, 127
sys_set_malloc
mpx_supt.c, 119
mpx_supt.h, 127
SYSTEM
pcb.h, 169
system.c
klogv, 100
kpanic, 101
system.h
asm, 68
cli, 68
GDT_CS_ID, 68
GDT_DS_ID, 68
hlt, 68
iret, 68
klogv, 70
kpanic, 70
no_warn, 68
nop, 68
NULL, 69
size_t, 69
sti, 69
u16int, 69
u32int, 69
u8int, 69
volatile, 69

table
index_table, 23
TABLE_SIZE
heap.h, 57
tables
page_dir, 28
tables.c
gdt_entries, 103
gdt_init_entry, 102
gdt_ptr, 103
idt_entries, 103
idt_ptr, 104
idt_set_gate, 102
init_gdt, 102
init_idt, 102
write_gdt_ptr, 103

write_idt_ptr, 103
tables.h
__attribute__, 53
access, 55
base, 55
base_high, 55
base_low, 55
base_mid, 55
flags, 56
gdt_init_entry, 53
idt_set_gate, 54
init_gdt, 54
init_idt, 54
limit, 56
limit_low, 56
sselect, 56
zero, 56
tables_phys
page_dir, 28
tail
aList, 7
ioqueue_s, 25
mList, 27
Queue, 34
test.c
er1, 223
er2, 223
er3, 223
er4, 223
er5, 223
erSize, 223
msg1, 224
msg2, 224
msg3, 224
msg4, 224
msg5, 224
msgSize, 224
proc1, 222
proc2, 222
proc3, 222
proc4, 222
proc5, 223
RC_1, 221
RC_2, 221
RC_3, 221
RC_4, 221
RC_5, 221
test.h
proc1, 225
proc2, 225
proc3, 225
proc4, 225
proc5, 226
time
alarm_s, 6
time.c
bcdToDec, 157
getTime, 157
parseTime, 158
setTime, 159
setTimeCall, 159
time.h
bcdToDec, 161
getTime, 161
parseTime, 161
setTime, 163
setTimeCall, 163
timeDaemon
alarm.c, 230
alarm.h, 234
titleSize
queue.c, 199
toupper
string.c, 115
string.h, 66
TRUE
mpx_supt.h, 124
type
CMCB_s, 8
LMCB_s, 26

u16int
system.h, 69
u32int
system.h, 69
u8int
system.h, 69
unblockPCB
processManagement.c, 183
processManagement.h, 195
unblockPCBCall
R2Commands.c, 207
R2Commands.h, 212
UP
comhand.c, 129
serial.c, 89
UP_ARROW
serial.c, 89
usermode
page_entry, 29

version
miscCommands.c, 151
miscCommands.h, 156
volatile
system.h, 69

WHITE
comhand.c, 129
WRITE
mpx_supt.h, 124
write_gdt_ptr
tables.c, 103
write_idt_ptr
tables.c, 103
writeable
page_entry, 30

year
    date_time, 12
yield
    context.c, 216
    context.h, 220

zero
    idt_entry_struct, 20
    tables.h, 56