Chapter 1

Introduction

Chapter 2

Toolbox Functions

2.1 Channel Functions

`sys_chan_read_b`

Prototype	**short** sys_chan_read_b(**short** channel)
Address	`0xFFE024`
channel	the number of the channel
Returns	the value read (if negative, error)

`sys_chan_read`

Prototype	**short** sys_chan_read(**short** channel, **unsigned char** ∗ buffer, **short** size)
Address	`0xFFE028`
channel	the number of the channel
buffer	the buffer into which to copy the channel data
size	the size of the buffer.
Returns	number of bytes read, any negative number is an error code

`sys_chan_readline`

Prototype	**short** sys_chan_readline(**short** channel, **unsigned char** ∗ buffer, **short** size)
Address	`0xFFE02C`
channel	the number of the channel
buffer	the buffer into which to copy the channel data
size	the size of the buffer
Returns	number of bytes read, any negative number is an error code

`sys_chan_write_b`

Prototype	**short** sys_chan_write_b(**short** channel, uint8_t b)
Address	`0xFFE030`
channel	the number of the channel
b	the byte to write
Returns	0 on success, a negative value on error

`sys_chan_write`

Prototype	**short** sys_chan_write(**short** channel, **const** uint8_t ∗ buffer, **short** size)
Address	`0xFFE034`
channel	the number of the channel
buffer	
size	
Returns	number of bytes written, any negative number is an error code

`sys_chan_status`

Prototype	**short** sys_chan_status(**short** channel)
Address	`0xFFE038`
channel	the number of the channel
Returns	the status of the device

`sys_chan_flush`

Prototype	**short** sys_chan_flush(**short** channel)
Address	`0xFFE03C`
channel	the number of the channel
Returns	0 on success, any negative number is an error code

`sys_chan_seek`

Prototype	**short** sys_chan_seek(**short** channel, **long** position, **short** base)
Address	`0xFFE040`
channel	the number of the channel
position	the position of the cursor
base	whether the position is absolute or relative to the current position
Returns	0 = success, a negative number is an error.

`sys_chan_ioctrl`

Prototype	**short** sys_chan_ioctrl(**short** channel, **short** command, uint8_t * buffer, **short** size)
Address	`0xFFE044`
channel	the number of the channel
command	the number of the command to send
buffer	pointer to bytes of additional data for the command
size	the size of the buffer
Returns	0 on success, any negative number is an error code

`sys_chan_open`

Prototype	**short** sys_chan_open(**short** dev, **const char** * path, **short** mode)
Address	`0xFFE048`
dev	the device number to have a channel opened
path	a ”path” describing how the device is to be open
mode	s the device to be read, written, both? (0x01 = READ flag, 0x02 = WRITE flag, 0x03 = READ and
Returns	the number of the channel opened, negative number on error

sys_chan_close

Prototype	**short** sys_chan_close(**short** chan)
Address	`0xFFE04C`
chan	the number of the channel to close
Returns	nothing useful

sys_chan_swap

Prototype	**short** sys_chan_swap(**short** channel1, **short** channel2)
Address	`0xFFE050`
channel1	the ID of one of the channels
channel2	the ID of the other channel
Returns	0 on success, any other number is an error

sys_chan_device

Prototype	**short** sys_chan_device(**short** channel)
Address	`0xFFE054`
channel	the ID of the channel to query
Returns	the ID of the device associated with the channel, negative number for error

2.2 Block Device Functions

`sys_bdev_register`

Prototype	**short** sys_bdev_register(p_dev_block device)
Address	`0xFFE05C`
device	pointer to the description of the device to register
Returns	0 on succes, negative number on error

`sys_bdev_read`

Prototype	**short** sys_bdev_read(**short** dev, **long** lba, uint8_t * buffer, **short** size)
Address	`0xFFE060`
dev	the number of the device
lba	the logical block address of the block to read
buffer	the buffer into which to copy the block data
size	the size of the buffer.
Returns	number of bytes read, any negative number is an error code

`sys_bdev_write`

Prototype	**short** sys_bdev_write(**short** dev, **long** lba, **const** uint8_t * buffer, **short** size)
Address	`0xFFE064`
dev	the number of the device
lba	the logical block address of the block to write
buffer	the buffer containing the data to write
size	the size of the buffer.
Returns	number of bytes written, any negative number is an error code

`sys_bdev_status`

Prototype	**short** sys_bdev_status(**short** dev)
Address	`0xFFE068`
dev	the number of the device
Returns	the status of the device

`sys_bdev_flush`

Prototype	**short** sys_bdev_flush(**short** dev)
Address	`0xFFE06C`
dev	the number of the device
Returns	0 on success, any negative number is an error code

`sys_bdev_ioctrl`

Prototype	**short** sys_bdev_ioctrl(**short** dev, **short** command, uint8_t * buffer, **short** size)
Address	`0xFFE070`
dev	the number of the device
command	the number of the command to send
buffer	pointer to bytes of additional data for the command
size	the size of the buffer
Returns	0 on success, any negative number is an error code

2.3 File System Functions

`sys_fsys_open`

Prototype	**short** sys_fsys_open(**const char** ∗ path, **short** mode)
Address	`0xFFE074`
path	the ASCIIZ string containing the path to the file.
mode	the mode (e.g. r/w/create)
Returns	the channel ID for the open file (negative if error)

`sys_fsys_close`

Prototype	**short** sys_fsys_close(**short** fd)
Address	`0xFFE078`
fd	the channel ID for the file
Returns	0 on success, negative number on failure

`sys_fsys_opendir`

Prototype	**short** sys_fsys_opendir(**const char** ∗ path)
Address	`0xFFE07C`
path	the path to the directory to open
Returns	the handle to the directory if ¿= 0. An error if ¡ 0

`sys_fsys_closedir`

Prototype	**short** sys_fsys_closedir(**short** dir)
Address	`0xFFE080`
dir	the directory handle to close
Returns	0 on success, negative number on error

`sys_fsys_readdir`

Prototype	**short** sys_fsys_readdir(**short** dir, p_file_info file)
Address	`0xFFE084`
dir	the handle of the open directory
file	pointer to the t_file_info structure to fill out.
Returns	0 on success, negative number on failure

`sys_fsys_findfirst`

Prototype	**short** sys_fsys_findfirst(**const char** ∗ path, **const char** ∗ pattern, p_file_info file)
Address	`0xFFE088`
path	the path to the directory to search
pattern	the file name pattern to search for
file	pointer to the t_file_info structure to fill out
Returns	the directory handle to use for subsequent calls if ¿= 0, error if negative

sys_fsys_findnext

Prototype	**short** sys_fsys_findnext(**short** dir, p_file_info file)
Address	`0xFFE08C`
dir	the handle to the directory (returned by fsys_findfirst) to search
file	pointer to the t_file_info structure to fill out
Returns	0 on success, error if negative

sys_fsys_get_label

Prototype	**short** sys_fsys_get_label(**const char** ∗ path, **char** ∗ label)
Address	`0xFFE090`
path	path to the drive
label	buffer that will hold the label... should be at least 35 bytes
Returns	0 on success, error if negative

sys_fsys_set_label

Prototype	**short** sys_fsys_set_label(**short** drive, **const char** ∗ label)
Address	`0xFFE094`
drive	drive number
label	buffer that holds the label
Returns	0 on success, error if negative

sys_fsys_mkdir

Prototype	**short** sys_fsys_mkdir(**const char** ∗ path)
Address	`0xFFE098`
path	the path of the directory to create.
Returns	0 on success, negative number on failure.

sys_fsys_delete

Prototype	**short** sys_fsys_delete(**const char** ∗ path)
Address	`0xFFE09C`
path	the path of the file or directory to delete.
Returns	0 on success, negative number on failure.

sys_fsys_rename

Prototype	**short** sys_fsys_rename(**const char** ∗ old_path, **const char** ∗ new_path)
Address	`0xFFE0A0`
old_path	he current path to the file
new_path	the new path for the file
Returns	0 on success, negative number on failure.

`sys_fsys_set_cwd`

Prototype	**short** sys_fsys_set_cwd(**const char** ∗ path)
Address	`0xFFE0A4`
path	the path that should be the new current
Returns	0 on success, negative number on failure.

`sys_fsys_get_cwd`

Prototype	**short** sys_fsys_get_cwd(**char** ∗ path, **short** size)
Address	`0xFFE0A8`
path	the buffer in which to store the directory
size	the size of the buffer in bytes
Returns	0 on success, negative number on failure.

`sys_fsys_load`

Prototype	**short** sys_fsys_load(**const char** ∗ path, uint32_t destination, uint32_t ∗ start)
Address	`0xFFE0AC`
path	the path to the file to load
destination	the destination address (0 for use file's address)
start	pointer to the long variable to fill with the starting address
Returns	0 on success, negative number on error

`sys_fsys_register_loader`

Prototype	**short** sys_fsys_register_loader(**const char** ∗ extension, p_file_loader loader)
Address	`0xFFE0B0`
extension	the file extension to map to
loader	pointer to the file load routine to add
Returns	0 on success, negative number on error

`sys_fsys_stat`

Prototype	**short** sys_fsys_stat(**const char** ∗ path, p_file_info file)
Address	`0xFFE0B4`
path	the path to the file to check
file	pointer to a file info record to fill in, if the file is found.
Returns	0 on success, negative number on error

2.4 Text System Functions

`sys_txt_set_mode`

Prototype	**short** sys_txt_set_mode(**short** screen, **short** mode)
Address	`0xFFE0E0`
screen	the number of the text device
mode	a bitfield of desired display mode options
Returns	0 on success, any other number means the mode is invalid for the screen

`sys_txt_set_xy`

Prototype	**void** sys_txt_set_xy(**short** screen, **short** x, **short** y)
Address	`0xFFE0E8`
screen	the number of the text device
x	the column for the cursor
y	the row for the cursor

`sys_txt_get_xy`

Prototype	**void** sys_txt_get_xy(**short** screen, p_point position)
Address	`0xFFE0EC`
screen	the number of the text device
position	pointer to a t_point record to fill out

`sys_txt_get_region`

Prototype	**short** sys_txt_get_region(**short** screen, p_rect region)
Address	`0xFFE0F0`
screen	the number of the text device
region	pointer to a t_rect describing the rectangular region (using character cells for size and siz
Returns	0 on success, any other number means the region was invalid

`sys_txt_set_region`

Prototype	**short** sys_txt_set_region(**short** screen, p_rect region)
Address	`0xFFE0F4`
screen	the number of the text device
region	pointer to a t_rect describing the rectangular region (using character cells for size and siz
Returns	0 on success, any other number means the region was invalid

`sys_txt_set_color`

Prototype	**void** sys_txt_set_color(**short** screen, **unsigned char** foreground, **unsigned char** background)
Address	`0xFFE0F8`
screen	the number of the text device
foreground	the Text LUT index of the new current foreground color (0 - 15)
background	the Text LUT index of the new current background color (0 - 15)

`sys_txt_get_color`

Prototype	**void** sys_txt_get_color(**short** screen, **unsigned char** * foreground, **unsigned char** * background)
Address	`0xFFE0FC`
screen	the number of the text device
foreground	the Text LUT index of the new current foreground color (0 - 15)
background	the Text LUT index of the new current background color (0 - 15)

`sys_txt_set_cursor_visible`

Prototype	**void** sys_txt_set_cursor_visible(**short** screen, **short** is_visible)
Address	`0xFFE100`
screen	the screen number 0 for channel A, 1 for channel B
is_visible	TRUE if the cursor should be visible, FALSE (0) otherwise

`sys_txt_set_font`

Prototype	**short** sys_txt_set_font(**short** screen, **short** width, **short** height, **unsigned char** * data)
Address	`0xFFE104`
screen	the number of the text device
width	width of a character in pixels
height	of a character in pixels
data	pointer to the raw font data to be loaded

`sys_txt_get_sizes`

Prototype	**void** sys_txt_get_sizes(**short** screen, p_extent text_size, p_extent pixel_size)
Address	`0xFFE108`
screen	the screen number 0 for channel A, 1 for channel B
text_size	the size of the screen in visible characters (may be null)
pixel_size	the size of the screen in pixels (may be null)

`sys_txt_set_border`

Prototype	**void** sys_txt_set_border(**short** screen, **short** width, **short** height)
Address	`0xFFE10C`
screen	the number of the text device
width	the horizontal size of one side of the border (0 - 32 pixels)
height	the vertical size of one side of the border (0 - 32 pixels)

sys_txt_set_border_color

Prototype	**void** sys_txt_set_border_color(**short** screen, **unsigned char** red, **unsigned char** green, **unsign**
Address	`0xFFE110`
screen	the number of the text device
red	the red component of the color (0 - 255)
green	the green component of the color (0 - 255)
blue	the blue component of the color (0 - 255)

sys_txt_put

Prototype	**void** sys_txt_put(**short** screen, **char** c)
Address	`0xFFE114`
screen	the number of the text device
c	the character to print

sys_txt_print

Prototype	**void** sys_txt_print(**short** screen, **const char** ∗ message)
Address	`0xFFE118`
screen	the number of the text device
message	the ASCII Z string to print

2.5 Interrupt Functions

sys_int_enable_all

Prototype	**void** sys_int_enable_all()
Address	`0xFFE004`

sys_int_disable_all

Prototype	**void** sys_int_disable_all()
Address	`0xFFE008`

sys_int_disable

Prototype	**void** sys_int_disable(**unsigned short** n)
Address	`0xFFE00C`
n	the number of the interrupt: n[7..4] = group number, n[3..0] = individual number.

sys_int_enable

Prototype	**void** sys_int_enable(**unsigned short** n)
Address	`0xFFE010`
n	the number of the interrupt

sys_int_register

Prototype	p_int_handler sys_int_register(**unsigned short** n, p_int_handler handler)
Address	`0xFFE014`
n	the number of the interrupt
handler	pointer to the interrupt handler to register
Returns	the pointer to the previous interrupt handler

sys_int_pending

Prototype	**short** sys_int_pending(**unsigned short** n)
Address	`0xFFE018`
n	the number of the interrupt: n[7..4] = group number, n[3..0] = individual number.
Returns	non-zero if interrupt n is pending, 0 if not

sys_get_info

Prototype	**void** sys_get_info(p_sys_info info)
Address	`0xFFE01C`
info	pointer to a s_sys_info structure to fill out

sys_int_clear

Prototype	**void** sys_int_clear(**unsigned short** n)
Address	`0xFFE020`
n	the number of the interrupt: n[7..4] = group number, n[3..0] = individual number.

2.6 General Functions

sys_proc_exit

Prototype	**void** sys_proc_exit(**short** result)
Address	`0xFFE000`
result	the code to return to the kernel

sys_proc_run

Prototype	**short** sys_proc_run(**const char** ∗ path, **int** argc, **char** ∗ argv[])
Address	`0xFFE0D8`
path	the path to the executable file
argc	the number of arguments passed
argv	the array of string arguments
Returns	the return result of the program

sys_text_setsizes

Prototype	**void** sys_text_setsizes(**short** chan)
Address	`0x000000`
chan	

sys_mem_get_ramtop

Prototype	uint32_t sys_mem_get_ramtop()
Address	`0xFFE0B8`
Returns	the address of the first byte of reserved system RAM (one above the last byte the user program can u

sys_mem_reserve

Prototype	uint32_t sys_mem_reserve(uint32_t bytes)
Address	`0xFFE0BC`
bytes	the number of bytes to reserve
Returns	address of the first byte of the reserved block

sys_time_jiffies

Prototype	uint32_t sys_time_jiffies()
Address	`0xFFE0C0`
Returns	the number of jiffies since the last reset

sys_rtc_set_time

Prototype	**void** sys_rtc_set_time(p_time time)
Address	`0xFFE0C4`
time	pointer to a t_time record containing the correct time

`sys_rtc_get_time`

Prototype	**void** sys_rtc_get_time(p_time time)
Address	`0xFFE0C8`
time	pointer to a t_time record in which to put the current time

`sys_kbd_scancode`

Prototype	uint16_t sys_kbd_scancode()
Address	`0xFFE0CC`
Returns	the next scan code from the keyboard... 0 if nothing pending

`sys_kbd_layout`

Prototype	**short** sys_kbd_layout(**const char** * tables)
Address	`0xFFE0D4`
tables	pointer to the keyboard translation tables
Returns	0 on success, negative number on error