PROSE KERNAL ROUTINES (last updated 29-08-2011)

To call a PROSE kernal routine, load register A with the routine label, set other CPU registers as appropriate for the required routine and do a **call.lil** to "prose_kernal". EG: To print a line of text:

```
ld hl,message_txt       ; string location
ld a,kr_print_string    ; desired kernal routine
call.lil prose_kernal   ; call PROSE kernal routine
```

File System:	Keyboard and Mouse:	Environment variables:
kr_mount_volumes	kr_wait_key	kr_set_envar
kr_get_device_info	kr_get_key	kr_get_envar
kr_check_volume_format	kr_get_key_mod_flags	kr_delete_envar
kr_change_volume	kr_get_keymap_location	
kr_get_volume_info	kr_set_mouse_window	Timer:
kr_format_device	kr_get_mouse_position	
kr_make_dir	kr_get_mouse_counters	kr_time_delay
kr_change_dir	kr_set_pointer	kr_set_timeout
kr_parent_dir	kr_get_joysticks	kr_test_timeout
kr_root_dir		kr_read_rtc
kr_delete_dir	Text / Display:	kr_write_rtc
kr_open_file		
kr_set_file_pointer	kr_print_string	Sound:
kr_set_load_length	kr_clear_screen	
kr_read_file	kr_plot_char	kr_play_audio
kr_erase_file	kr_set_pen	kr_disable_audio
kr_rename_file	kr_get_pen	
kr_create_file	kr_background_colours	Memory:
kr_write_file	kr_set_cursor_position	
kr_get_total_sectors	kr_draw_cursor	kr_get_mem_base
kr_dir_list_first_entry	kr_remove_cursor	kr_get_mem_top
kr_dir_list_get_entry	kr_set_cursor_image	kr_allocate_ram
kr_dir_list_next_entry	kr_scroll_up	kr_deallocate_ram
kr_read_sector	kr_os_display	
kr_write_sector	kr_get_video_mode	Misc:
kr_file_sector_list	kr_set_video_mode	
kr_get_dir_cluster	kr_get_charmap_addr_xy	kr_get_version
kr_set_dir_cluster	kr_get_cursor_position	kr_dont_store_registers
kr_get_dir_name	kr_wait_vrt	
kr_get_disk_sector_ptr	kr_char_to_font	
Serial Comms:	Strings:	
kr_serial_receive_header	kr_compare_strings	
kr_serial_receive_file	kr_hex_byte_to_ascii	
kr_serial_send_file	kr_ascii_to_hex_word	
kr_serial_tx_byte	kr_get_string	
kr_serial_rx_byte		

Notes:

- When a Z80 mode program calls the PROSE kernal, location pointer registers such as HL in the example above automatically have their MSB [23:16] set to the MBASE register by the Kernal routine.

- All DISK routines set the zero flag on return if the operation was successful. If the zero flag is not set, CPU register "A" will contain an error code (see PROSE manual for list). Other routines set the flags as shown below.

- IX and IY are preserved by all kernal routines, assume all other registers are trashed unless otherwise stated.

---

kr_mount_volumes

- Action: Rescans hardware for storage devices.
- Input : E (0= Show device list, 1 = Mount quietly)
- Output: None

---

kr_get_device_info

- Action: Returns info about the storage devices
- Input : None
- Output: HL = Location of device info table
  DE = Location of driver table
  B = Device count
  A = Currently selected driver

---

kr_check_volume_format

- Action: Checks if currently selected volume is formatted to FAT16
- Input : None
- Output: See error codes

---

kr_change_volume

- Action: Changes volume selection
- Input : E = Desired volume number
- Output: See error codes

---

kr_get_volume_info

- Action: Returns info about storage volumes.
- Input : None
- Output: HL =location of volume mount list (see PROSE manual for info)
  B = volume count
  A = currently selected volume

---

kr_format_device

- Action: Formats a device to FAT16 (no MBR is created and the entire device is treated as one volume, max 2GB)
- Input : E = device
  HL = location of desired volume label
- Output: See error codes

---

kr_make_dir

- Action: Creates a new directory
- Input : HL = Location of zero-terminated dir name
- Output: See error codes

---

kr_change_dir

- Action: Changes current directory
- Input : HL = Location of zero-terminated dir name
- Output: See error codes

---

kr_parent_dir

- Action: Moves towards the root directory by one place
- Input : None
- Output: See error codes

---

kr_root_dir

- Action: Sets the root dir as the current directory
- Input : None
- Output: None

---

kr_delete_dir

- Action: Removes an (empty) directory
- Input : HL = Location of zero-terminated dir name
- Output: See error codes

---

kr_open_file

- Action: Opens a file so that data from it may be loaded.
- Input : HL = Location of zero-terminated file name
- Output: If zero flag set:

  HL = start cluster of file
  C:DE = length of file (32 bit)

  Else: see error codes

---

kr_set_file_pointer

- Action: Moves the file pointer to a position within the currently opened file
- Input : C:DE (32 bit file pointer)
- Output: None

---

kr_set_load_length

- Action: Sets the maximum data transfer length for a file read
- Input : DE = load length (24 bit)
- Output: None

---

kr_read_file

- Action: Reads data from the currently opened file
- Input : HL = load address
- Output: see error codes

---

kr_erase_file

- Action: Deletes a file
- Input : HL = Location of zero-terminated file name
- Output: See error codes

---

kr_rename_file

- Action: Renames a file or directory
- Input : HL = Location of original name
  DE = Location of new name
  (Both strings should be zero terminated.)
- Output: See error codes

---

kr_create_file

- Action: Creates a new file (zero bytes) for writing data to.
- Input : HL = location of zero-terminated file name
- Output: See error codes

---

kr_write_file

- Action: Appends data to an existing file
- Input : HL = location of zero terminated filename
  DE = source address of data
  BC = length (24 bit)
- Output: See error codes

---

kr_get_total_sectors

- Action: Returns the total capacity (in sectors) of the currently selected volume
- Input : None
- Output: DE = sector count (24 bit)

---

kr_dir_list_first_entry

- Action: Returns the first line of a directory
- Input : None
- Output: If zero flag set:

```
HL    = Location of null terminated filename string
C:DE  = Length of file (if applicable)
B     = File flag 0 = File, 1 = Dir
```

Else: see error codes

---

kr_dir_list_get_entry

- Action: Returns a line from directory
- Input : None
- Output: If zero flag set:

```
HL     = Location of null terminated filename string
C:DE   = Length of file  (if applicable)
B      = File flag 0 = File, 1 = Dir
```

Else: see error codes

---

kr_dir_list_next_entry

- Action: Returns next line of directory
- Input : None
- Output: If zero flag set:

```
HL     = Location of null terminated filename string
C:DE   = Length of file  - if applicable)
B      = File flag 0 = File, 1 = Dir
```

Else: see error codes

---

kr_read_sector

- Action: Reads a sector from the specified device to the target address
- Input : HL = destination address
  C:DE = sector (32 bit)
  B = device number
- Output: See error codes

---

kr_write_sector

- Action: Writes a sector from specified address to the specified device
- Input : HL = source address
  C:DE = sector (32 bit)
  B = device number
- Output: See error codes

---

kr_file_sector_list

- Action: Used to obtain a list of the sectors that a file occupies
- Input : HL = cluster
  E = sector within cluster
- Output: E = Next sector offset
  HL = Updated cluster
  BC = Location of variable holding 32 bit sector (LSB)
  [Also see error codes]

Notes: First call "kr_open_file" with the filename in HL as normal. On exit, HL will be equal the first cluster that the file occupies. Clear E (as we're starting at the first sector of a cluster), and call "kr_file_sector_list". On return E and HL are updated to the values required for the next time the routine is called and BC points to the lowest significant byte of the sector address (LBA0). Copy the 4 bytes from BC to BC+3 to your sector list buffer and loop around, calling "kr_file_sector_list" for as many sectors as are used by the file (simply subtract 512 from a variable holding the file size every call until variable is = < 0)

---

kr_get_dir_cluster

- Action: Reads the cluster location of the current directory
- Input : none
- Output: DE = current dir's cluster address

---

kr_set_dir_cluster

- Action: Sets the current directory cluster location
- Input : DE = cluster address to set as current dir
- Output: none

---

kr_get_dir_name

- Action: Returns current directory name
- Input : None
- Output: HL = location of directory name ASCII string
  [See also error codes]

---

kr_get_disk_sector_ptr

- Action: Returns location of LSB of LBA sector variable and location of sector buffer for external drivers
- Input : none
- Output: HL = location of LSB of 32-bit sector address variable
  DE = location of sector buffer

---

kr_wait_key

- Action: Pauses until a key is pressed
- Input : None
- Output: A = Scancode of keypress
  B = ASCII character as defined by keymap and modified by shift/alt/CRTL as applicable. B = 0 if no applicable ASCII data.

---

kr_get_key

- Action: Returns any keypresses that are in the buffer (does not wait)
- Input : None
- Output: If zero flag is set:

```
A  = New scancode
B  = ASCII character as defined by keymap and modified
     by shift/alt/CRTL as applicable. B = 0 if no applicable
     ASCII data.

Else: No new key data is in the buffer
```

---

kr_get_key_mod_flags

- Action: Returns the status of the modifier keys
- Input : None
- Output: A, bit 0 - left shift
  1 - left/right ctrl
  2 - left GUI
  3 - left/right alt
  4 - right shift
  5 - right GUI
  6 - Apps

---

kr_serial_receive_header

- Action: Waits for a file header from serial port
- Input : HL = location of zero-terminated filename
  E = timeout allowance in seconds
- Output: If zero flag set, all OK, xDE returns location of serial file header

```
Else, A =   $83 : timed out error
            $84 : memory address out of range
            $85 : comms error
            $86 : checksum bad
            $87 : Incorrect file
```

---

kr_serial_receive_file

- Action: Waits for a serial file transfer following the reception of a header
- Input : HL = load address
- Output: If zero flag is set, all OK.

```
Else, A =   $84 : memory address out of range,
            $85 : comms error
            $86 : checksum error
```

---

kr_serial_send_file

- Action: Sends a file to the serial port
- Input : HL = filename location
           DE = source address
           BC = length
- Output: If zero flag is set, all OK.

```
Else, A=  $81 = Save length zero,
          $84 = memory address out of range
          $85 = comms error
```

---

kr_serial_tx_byte

- Action: Sends a single byte to the serial port
- Input : E = byte to send
- Output: None

---

kr_serial_rx_byte

- Action: Waits for a single byte from serial port
- Input : E = timeout allowance in seconds
- Output: Zero Flag set if byte received OK, Else operation timed out (A = $83)

---

kr_print_string

- Action: Writes a string of text to the display at the current cursor position and in the current pen colour
- Input : HL (Zero-terminated ASCII string location)
- Output: HL = address of the string terminating zero + 1
  (BC and DE are preserved)

---

kr_clear_screen

- Action: Clears the display window
- Input : None
- Output: None

---

kr_wait_vrt

- Action: Waits for the Vertical Retrace flip-flop to become set (ie: the last scanline of display)
- Input : None
- Output: None

---

kr_set_cursor_position

- Action: Moves the cursor position to a specific place
- Input : B = x char coord
  C = y char coord
- Output: Zero flag is set if coordinates are within the display window, unset if not.

---

kr_plot_char

- Action: Plots a character at a specific location in current pen colour (doesn't affect cursor position)
- Input : B = x char coord
  C = y char coord
  E = ASCII char to plot
- Output: Zero flag is set if coordinates are within the display window, unset if not.

---

kr_set_pen

- Action: Changes the current pen colour
- Input : E = pen colour:

  Bits [7:4] = background colour selection
  [3:0] = character's pixel colour selection.
- Output: None

---

kr_background_colours

- Action: Changes the palette of 16 colours used by the OS
- Input : HL (location of colour data, standard 16 words of "0RGB" format)
- Output: None

---

kr_draw_cursor

- Action: Draws the cursor image as defined by the routine "kr_set_cursor_image"
- Input : None
- Output: None

---

kr_get_pen

- Action: Returns the current pen colour
- Input : None
- Output: E = pen colour:

```
Bits [7:4] = background colour selection
     [3:0] = character's pixel colour selection.
```

---

kr_scroll_up

- Action: Scrolls the display up a line
- Input : None
- Output: None

---

kr_os_display

- Action: Restores the display hardware settings to that used by the OS
  (useful only if the area of VRAM used by PROSE were not overwritten.)
- Input : None
- Output: None

---

kr_get_video_mode

- Action: Returns the OS video mode and size of the OS window (in characters)
- Input : None
- Output:
  A = Video Mode
  B = Columns
  C = Lines

---

kr_get_charmap_addr_xy

- Action: Returns address of character map and attribute map for a specific coordinate

- Input : B = x coord
  C = y coord

- Output: HL = OS character map address
  DE = OS attribute map addresss

---

kr_get_cursor_position

- Action: Returns cursor position

- Input : None

- Output: B = x char coord
  C = y char coord

---

kr_set_video_mode

- Action: Sets the Video Mode

  Input: E= 0 : 80x60
  1 : 80x30
  2 : 40x60
  3 : 40x30

- Output: Zero Flag set if all OK. Error code 88h if A is out of range.

---

kr_set_cursor_image

- Action: Selects which character is to be used for the cursor.

- Input : E: ASCII character to use (EG: 05fh = underscore, 07fh = solid block)

- Output: None.

---

kr_remove_cursor

- Action: Removes the cursor image from the character map (replaces with character that was saved by kr_draw_cursor)
- Input : None
- Output: None

---

kr_char_to_font

- Action: Patches the PROSE font (allows user defined characters)
- Input : E = ASCII character to change
  HL = address of font data (8 bytes)
- Output: None

---

kr_set_envar

- Input : HL = location of zero-terminated variable name string
  DE = location of zero-terminated variable value string
- Output: ZF set if OK

---

kr_get_envar

- Input : HL = location of zero-terminated variable name string
- Output: ZF set if name found
  DE = location of zero-terminated variable value string

---

kr_delete_envar

- Input : HL = location of zero-terminated variable name string
- Output: ZF set if OK

---

kr_set_mouse_window

- Action: Sets the constraining dimensions for absolute mouse pointer position
- Input : HL = width of window in pixels
  DE = height of window in pixels
- Output: None

---

kr_get_mouse_position

- Action: Returns the absolute position of the mouse pointer within the window
  button status and scroll wheel counter (if applicable)
- Input: None
- Output: If zero flag is set:

  HL = x coord
  DE = y coord
  A = buttons. Bit 0 = left, bit 1 = right, bit 2 = middle
  B = mouse wheel counter (always zero if no mouse wheel)

  If zero Flag is not set mouse is not enabled

---

kr_get_mouse_counters (previously "kr_get_mouse_motion")

- Action: Returns the current value of the (wrap around) mouse counters
  button status and scroll wheel counter (if applicable)
- Input : None
- Output: If zero flag is set:

  HL = x coord
  DE = y coord
  A = buttons. Bit 0 = left, bit 1 = right, bit 2 = middle
  B = mouse wheel counter (always zero if no mouse wheel)

  If zero Flag is not set mouse is not enabled

---

kr_time_delay

- Action: Pauses (Granularity of 30 microseconds)
- Input : DE = 32768Hz ticks to pause. (Max value 65535, IE: 2 seconds)
- Output: None

---

kr_compare_strings

- Action: Compares ASCII strings, ignoring the case.

- Input: HL = location of string 1
  DE = location of string 2
  B = count of characters to compare

- Output: Zero flag set if strings are the same

---

kr_hex_byte_to_ascii

- Action: Puts ascii version of hex byte at (HL) and (HL+1)

- Input : HL = Desired location for ASCII output
  E = byte to convert

- Output: HL=HL+2

---

kr_ascii_to_hex_word

- Action: Converts ASCII hex characters to 24 bit hexadecimal number

- Input: HL = location of ASCII string (Note: routine scans for first non-space character)

- Output: If zero flag set:

  DE = result

  Else, A = $81: No ASCII chars
  $82: Bad ASCII chars

---

kr_get_string

- Action: Waits for user to enter a string of characters followed by return (Escape quits)

- Input : HL = location of string
  E = max number of characters.

- Output: If zero flag set, all OK:

  A = number of characters entered.

  Else: A = $80 if ESC was pressed
  $81 if no characters were entered

---

kr_get_version

- Action: Returns version info for OS and Hardware
- Input : None
- Output: HL = Version of PROSE OS
  DE = version of AMOEBA FPGA config

---

kr_dont_store_registers

- Action: Prevents the OS caching the contents of the CPU registers when an app returns control to it.
- Input : None
- Output: None

---

kr_read_rtc

- Action: Reads the real time clock data
- Input : None
- Output: HL = location of ez80 time data:

  (sec, min, hr, d.o,w, date, mon, year, century)

---

kr_write_rtc

- Action: Writes data to the real time clock
- Input : HL = location of ez80 time data

  (sec, min, hr, d.o.w, date, mon, year, century)
- Output: None

---

kr_get_keymap_location

- Action: Returns location of keymap within the OS
- Input : None
- Output: HL = location of keymap data within OS

---

kr_get_mem_base

- Action: Returns the first available address (IE: not used by the OS) in system RAM, VRAM A and VRAM B
- Input : none
- Output: HL = first free system RAM address
  DE = same for VRAM A
  BC = same for VRAM B

---

kr_play_audio

- Action: Plays sound samples
- Input : HL = Address of sound description table
  C = channel enable bits.
- Output: None

Notes: The sound description table format is:

```
 00h [3 bytes]    ;location (address in VRAM B)
 03h [3 bytes]    ;length in bytes
 06h [3 bytes]    ;loop location (address in VRAM B)
 09h [3 bytes]    ;loop length in bytes
 0ch [2 bytes]    ;period constant (see hardware manual)
 0eh [1 bytes]    ;volume (0-64)
```

The channel enable bits set in C specify which channels are to play the sound:

```
 Bit 0 = channel 0
 Bit 1 = channel 1
 Bit 2 = channel 2
 etc etc
```

---

kr_disable_audio

- Action: Silences all channels by disabling the audio hardware and silencing each channel's volume register
- Input : none
- Output: none

---

kr_get_joysticks

- Action: Reads status of joystick pins
- Input : none
- Output: E = joystick 0, D = joystick 1

```
Bit 0 = Up
Bit 1 = Down
Bit 2 = Left
Bit 3 = Right
Bit 4 = Fire 0
Bit 5 = Fire 1
```

Note: Bit set = direction / button asserted.

---

kr_set_timout

- Action: Sets the timeout period (in 32768 Hz ticks), max 65536 (two seconds) and starts the countdown
- Input : DE = time-out value
- Output: none

---

kr_test_timeout

- Action: Reports on status of time out flag
- Input : none
- Output: ZF not set if timed out (no time-out if zero flag set)

---

kr_set_pointer

- Action: Initializes + enables / disables the mouse pointer sprite.

- Input: E: 1 = Enable pointer, 0 = disable pointer
  D: 1 = Use default PROSE pointer (no other registers required)
  0 = Use custom pointer, the following parameters are required:

  HL = location of sprite data in system memory (copied to end of sprite RAM by this routine). Sprite data consists of the definition pixel data followed by:

  $xx (byte) - first palette index used by sprite
  $xx (byte) - number of colours (words) in palette data

  then.. palette data

  C = pointer sprite height in lines (max 32)
  B = palette 0-3 to use for pointer (and all) sprites

- Output: Returns with Zero Flag not set if mouse driver was not activated.

---

kr_allocate_ram

- Action: Sets aside an area at the top of memory for a specific use

- Input : BC = number of bytes to allocate
  : E = 0 : allocate system RAM
  1 : allocate VRAM_A (Video RAM)
  2 : allocate VRAM_B (Sprite/Audio RAM)

  Output: HL = address of allocated RAM, Zero Flag set if allocated OK

Notes : Apps only need to allocate memory if it is to be protected from program loads etc after the app has exited to PROSE. EG: driver code. Otherwise apps can assume they have the entire (non-protected) RAM range to themselves, checking the free location range with "kr_get_mem_base" and "kr_get_mem_top"

---

kr_deallocate_ram

- Action: Releases an area of RAM. IE: Moves the current allocation pointer upwards.

- Input : BC = number of bytes to deallocate
  E = 0 : Deallocate system RAM,
  = 1 : deallocate VRAM_A (Video RAM)
  = 2 : deallocate VRAM_B (Sprite/Audio RAM)

- Output: None

---

kr_get_mem_top

- Action: Returns the maximum free address locations in system RAM, VRAM A and VRAM B

- Input : none

- Output: HL = system RAM high
  DE = VRAM_A high (Video RAM)
  BC = VRAM_B high (Sprite/Audio RAM)
