YOKOI TRANSLATION ARCHIVE / 001

Wonder Witch Technical Manual

Technical English translation

Original first edition / interim publication / January 24, 2002

Published by Digitalis / written by Atsushi Watanabe

ABOUT THIS EDITION

A functional translation of all sections in the supplied 81-page Japanese manual.
Repeated BIOS table labels are compacted; contracts, warnings, and conflicts are retained.
Use as a research reference. The 2002 source itself warns that some claims were untested.

READ / VERIFY / BUILD

Contents

The printed-page references follow the original Japanese manual. This PDF is a compact English technical edition, so its pagination differs.

Wonder Witch Technical Manual - technical English translation

First edition, interim publication dated January 24, 2002. Published by Digitalis; written by Atsushi Watanabe (A. Watanabe / S.W.).

This is a functional technical translation of all sections of the supplied 81-page Japanese manual. Repeated labels such as "arguments," "return value," and "operation" are compressed into tables. Page references are printed-page numbers; PDF pages are one greater after the cover.

0. Preface (printed pages 1-2)

The bundled WonderWitch Developer's Manual was written for ordinary C programmers and contains almost no low-level hardware information. That makes it difficult for microcomputer engineers who work in machine code or assembly. This book is intended as a lower-layer technical reference with enough detail to develop in C, machine code, or assembly.

WonderWitch is a consumer development kit with substantial constraints. Even engineers who avoid closed-source software are asked to use its BIOS and OS. This manual does not document the hardware directly. Early work used reverse engineering, but that material was removed after the manufacturer said it violated the software license. The remaining information came from the bundled manuals, manufacturer questions, and experiments intended to clarify missing or ambiguous behavior.

There was not enough time or space to write sample programs for the book. The author calls this a weakness to be improved later and supplied working material on floppy disk. The intended reader understands the Intel 8086 family and assembly. Reading this beside the Developer's Manual should give a deeper view of WonderWitch.

1. Contents (printed page 3)

The manual covers system organization; memory, I/O, and interrupts; FreyaBIOS key, display, text, serial, sound, timer, system, and bank services; FreyaOS processes, filesystem, indirect libraries, and C libraries; and the supplied assembly material.

2. System organization (printed pages 4-6)

2.1 Predicted memory map

The author emphasizes that this map is inferred from header files and the NEC V30MZ specification, not guaranteed manufacturer documentation.

Physical range	Purpose
`00000h-03FFFh`	16 KiB console IRAM, used by BIOS, stack, and display data
`10000h-1FFFFh`	Mapped 64 KiB SRAM bank
`20000h-2FFFFh`	Mapped ROM0 bank
`30000h-3FFFFh`	Mapped ROM1 bank
`40000h-7FFFFh`	Shown as unused/unknown
`80000h-EFFFFh`	Flash banks 0-6 / linear ROM0 area
`F0000h-FFFFFh`	FreyaBIOS, flash bank 7

The mapped flash windows are `20000h-3FFFFh`; physical flash banks 0-7 occupy `80000h-FFFFFh`. The mapped SRAM window is `10000h-1FFFFh`. SRAM bank roles are:

Bank	Role
0	FreyaOS soft filesystem (`/ram0`)
1	FreyaOS user process 2
2	FreyaOS user process 1
3	FreyaOS work/process area

All areas not explicitly identified as console memory are on the cartridge.

2.2 I/O

I/O details were not public. Applications are expected to use BIOS services.

2.3 Interrupts

BIOS calls the registered user handler with a far call, so it must return with `retf`; a C handler must be a far function. BIOS saves general registers plus DS and ES and issues EOI. DS is loaded from the value supplied when the handler was registered. ES is unspecified because C may use it for far pointers. These rules do not apply when bypassing FreyaBIOS.

The manufacturer did not recommend a private interrupt stack. FreyaBIOS 1.0 has a hook-related bug covered by later FreyaOS/libww support, so C programs should use the functions declared by `sys/system.h`.

Hardware interrupt vectors, lowest to highest priority:

Vector	ID	Trigger	Meaning
`28h`	0	level	serial transmit ready
`29h`	1	edge	key press
`2Ah`	2	level	cartridge / RTC alarm
`2Bh`	3	level	serial receive ready
`2Ch`	4	edge	display line match
`2Dh`	5	edge	VBlank timer count-up
`2Eh`	6	edge	start of VBlank
`2Fh`	7	edge	HBlank timer count-up

While `/ram0` is being accessed, SRAM bank 0 is mapped and the user process's SRAM is not visible. An interrupt that touches user SRAM without switching the bank can corrupt a `/ram0` file. In a normal process, obtain the process SRAM bank with `pcb_get_srambank()`, save the currently mapped bank if different, map the process bank, perform the handler work, and restore the original bank.

An indirect library cannot trust `pcb_get_srambank()` because of an IL bug. Two suggested strategies are: defer handler work whenever `/ram0` is active; or, when installing the handler, allocate IRAM with `SYS_ALLOC_IRAM`, save the process SRAM bank there, and recover it in the handler with `SYS_GET_MY_IRAM`.

FreyaBIOS vectors are `10h` EXIT, `11h` KEY, `12h` DISP, `13h` TEXT, `14h` COMM, `15h` SOUND, `16h` TIMER, `17h` SYSTEM, and `18h` BANK. FreyaOS uses `30h` PROCESS, `31h` FILESYS, and `32h` ILIB; their direct syscall details were not public.

3. Hardware and FreyaBIOS (printed pages 7-56)

FreyaBIOS lives in the protected portion of cartridge flash and can be replaced through the monitor for future updates. Hardware access is expected to go through BIOS, OS, or the supplied libraries. FreyaBIOS 1.0 predates WonderSwan Color; Color support is supplied by FreyaOS and `libwwc.lib`.

The console CPU is a NEC V30MZ at 3.072 MHz, with a 20-bit address bus, 16-bit data/I/O bus, pipelining, and a CMOS static design able to stop the internal clock completely. The cartridge has 512 KiB flash (64 KiB used by BIOS) and 256 KiB SRAM. Console IRAM is 16 KiB and is mainly used by BIOS and the stack.

3.1 Exit - INT `10h` (printed page 8)

`INT 10h` terminates the program. Normal startup code begins at file offset `0000h`, calls C `main`, and invokes this service after `main` returns, so an ordinary assembly main routine may simply `ret`. Code that omits FreyaOS or the normal startup must call `INT 10h` itself.

3.2 Keys - INT `11h` (printed pages 9-11)

The hardware scans three groups: X1-X4, Y1-Y4, and A/B/Start. They cannot be read simultaneously at the hardware level. FreyaBIOS cycles the groups from a VBlank-timer handler, caches them, and implements repeat from the previous sample. Sound and Color power buttons are not exposed.

Key mask: bit 1 Start, bit 2 A, bit 3 B, bits 4-7 X1/up, X2/right, X3/down, X4/left, and bits 8-11 Y1/up, Y2/right, Y3/down, Y4/left.

AH	Service	Inputs	Result / behavior
`00h`	`KEY_PRESS_CHECK`	none	AX = keys currently held; no repeat
`01h`	`KEY_HIT_CHECK`	none	AX = newly pressed keys
`02h`	`KEY_WAIT`	none	wait, then AX = pressed keys
`03h`	`KEY_SET_REPEAT`	BL interval, BH initial delay	units approximately 1/75 second
`04h`	`KEY_GET_REPEAT`	none	AL interval, AH delay
`05h`	`KEY_HIT_CHECK_WITH_REPEAT`	none	AX = key hits with repeat

3.3 Display - INT `12h` (printed pages 12-24)

The virtual screen is 256 x 256 pixels and the LCD viewport is 224 x 144. The horizontal period is 256 dots with 32 blanked; the vertical period is 159 lines with 15 blanked, producing approximately 75.47 Hz (`12000 / 159`). Screen edges wrap. Tiles are 8 x 8. Mono supports 512 tile definitions and four colors per tile selected from eight display shades. Color supports 1024 tiles and 16 colors per tile from 4096 colors.

There are two independently scrollable screens and 128 sprites, with at most 32 sprites on one line. Screen 2 and sprites have pixel-coordinate clipping windows. The stated priority chain is Screen 1, low-priority sprite, Screen 2, high-priority sprite, then border color when everything is transparent. Color sprite palettes are shared with screen palettes 8-15.

Startup variants `ASCII1`, `ASCII2`, `JPN1`, and `JPN2` determine whether Screen 1 and Japanese text are available. Four-color Color mode can use tiles through 1023; 16-color modes can use all 0-1023. Startup normally selects grayscale, configures VRAM, selects Screen 2 as the text screen, disables all windows and sprites, and sets the enabled screens appropriate to the startup variant. Color software must detect hardware and select its color mode first.

Tile word: bits 0-8 tile index; bits 9-12 palette; bit 13 Color tile-index bit 9; bit 14 horizontal flip; bit 15 vertical flip. For sprites, bits 9-11 select palette 8-15 using the value minus 8, bit 12 is priority over Screen 2, bit 13 selects outside rather than inside the sprite window, and bits 14-15 flip.

AH	Service	Register contract / notes
00h	DISPLAY_CONTROL	enable Screen 1, Screen 2, sprites, sprite window; Screen 2 window mode; mono border color and Color border palette
01h	DISPLAY_STATUS	returns display-control state
02h	FONT_SET_MONODATA	BX start, CX count, DS:DX 8 bytes/tile; expands foreground/background selected by AH=05h; invalid in 16-color modes
03h	FONT_SET_COLORDATA	BX start, CX count, DS:DX 16 bytes/tile planar 2bpp; invalid in 16-color modes
04h	FONT_GET_DATA	BX start, CX count, DS:DX output; invalid in 16-color modes
05h	FONT_SET_COLOR	BX bits 0-1 foreground, 2-3 background
06h	FONT_GET_COLOR	AX returns that mapping
07h	SCREEN_SET_CHAR	AL screen; BL/BH x/y; CL/CH width/height; DS:DX tile words
08h	SCREEN_GET_CHAR	same rectangle, output to DS:DX
09h	SCREEN_FILL_CHAR	rectangle plus DX tile word
0Ah	SCREEN_FILL_ATTR	rectangle, DX value, SI mask; writes (old AND SI) OR DX
0Bh	SPRITE_SET_RANGE	BX first, CX count; only that contiguous range is visible; first + count must be at most 128
0Ch/0Dh	SPRITE_SET/GET_CHAR	BX sprite, CX input or AX output attribute word
0Eh/0Fh	SPRITE_SET/GET_LOCATION	BX sprite; DH/DL y/x input or AH/AL output
10h/11h	SPRITE_SET/GET_CHAR_LOCATION	combined attributes and location; getter returns AX plus DX
12h	SPRITE_SET_DATA	BX first, CX count, DS:DX array of four-byte sprite records; record byte order places Y before X
13h/14h	SCREEN_SET/GET_SCROLL	AL screen; BH/BL y/x input, AH/AL output, in pixels
15h/16h	SCREEN2_SET/GET_WINDOW	BH/BL y/x and CH/CL height/width; getter returns AX and DX
17h/18h	SPRITE_SET/GET_WINDOW	same format as Screen 2 window
19h/1Ah	PALETTE_SET/GET_COLOR	BX palette 0-15; CX input or AX output packs four 4-bit LCD-color selectors; mono only
1Bh/1Ch	LCD_SET/GET_COLOR	CX:BX input / DX:AX output packs eight 4-bit shade levels, 0 white through 15 black; mono only
1Dh/1Eh	LCD_SET/GET_SEGMENTS	LCD icon/orientation bits
1Fh/20h	LCD_SET/GET_SLEEP	bit 0: 0 display on, 1 display off/sleep
21h	SCREEN_SET_VRAM	AL screen, BL encoded VRAM address; unpublished, normally startup-owned
22h	SPRITE_SET_VRAM	BL encoded VRAM address; unpublished, normally startup-owned

The manual's LCD-segment table labels bit 1 horizontal and bit 2 vertical, but its supplied DISP.INC, current hardware definitions, and AthenaBIOS agree on bit 1 vertical and bit 2 horizontal. The latter is treated as authoritative.

3.4 Text - INT 13h (printed pages 25-31)

Either screen can be the text screen. A whole screen or rectangle can be initialized. ASCII mode consumes one tile per defined one-byte glyph; mixed or Shift-JIS modes consume one tile per text cell. The modes are TEXT_MODE_ANK 0 (one-byte only), TEXT_MODE_ANK_SJIS 1 (one- and two-byte), and TEXT_MODE_SJIS 2 (two-byte; BIOS converts one-byte ASCII to Shift-JIS). BIOS text services must not be used in 16-color or packed Color modes.

AH	Service	Register contract / notes
00h	TEXT_SCREEN_INIT	initializes 28 x 18 at 0,0; ASCII base is 512 minus defined glyph count, other modes use base 8
01h	TEXT_WINDOW_INIT	BL/BH x/y, CL/CH width/height, DX base tile
02h/03h	TEXT_SET/GET_MODE	BX input; implementation returns mode in AX
04h	TEXT_PUT_CHAR	BL/BH x/y, CX Shift-JIS code
05h	TEXT_PUT_STRING	BL/BH x/y, DS:DX NUL-terminated string
06h	TEXT_PUT_SUBSTRING	same plus CX character count
07h	TEXT_PUT_NUMERIC	BL/BH x/y, CL width, CH flags, DX value; bit 0 hex, bit 1 zero pad, bit 2 alignment, bit 3 signed, bit 7 write to buffer
08h	TEXT_FILL_CHAR	BL/BH x/y, CX length, DX character
09h/0Ah	TEXT_SET/GET_PALETTE	BX input / AX output
0Bh	TEXT_SET_ANK_FONT	BL first code, BH 0 mono or 1 four-color, CX count, DS:DX font
0Ch	TEXT_SET_SJIS_FONT	BX:DX far resolver; resolver takes AX Shift-JIS, returns CS:SI font and carry on miss; BX=0 disables two-byte text
0Dh	TEXT_GET_FONTDATA	CX code, DS:DX eight-byte output; AX 0 success or nonzero invalid code
0Eh/0Fh	TEXT_SET/GET_SCREEN	AL input / AX output
10h/11h	CURSOR_DISPLAY/STATUS	AL 0 hidden or 1 visible; status also reports blink visibility
12h/13h	CURSOR_SET/GET_LOCATION	x/y/width/height in BL/BH/CL/CH; getter AX/DX
14h/15h	CURSOR_SET/GET_TYPE	BL palette, CL blink period; default palette 1 and period 30

The printed bit-2 numeric-alignment labels are reversed relative to the current SDK and AthenaBIOS. AthenaBIOS implements bit 2 as "align left"; zero aligns right. The manual also says TEXT_GET_MODE returns BX, while the open handler returns AX. No SDK change was made for either disputed line.

3.5 Serial - INT 14h (printed pages 32-36)

The port is asynchronous full duplex, 8 data bits, 1 stop bit, at 9600 or 38400 bps. There is timeout, receive-overrun detection, key cancellation, and an unpublished XMODEM path. There is no XON/XOFF or modem control.

Errors: 0000h OK; 8100h busy; 8101h timeout; 8102h overrun; 8103h cancelled; 8104h invalid XMODEM state; 8105h XMODEM cancelled; 8106h block lost; 8107h too large.

AH	Service	Register contract / notes
00h/01h	COMM_OPEN/CLOSE	set baud before opening
02h	COMM_SEND_CHAR	BL byte; AX status
03h	COMM_RECEIVE_CHAR	AX byte or negative/error code
04h	COMM_RECEIVE_WITH_TIMEOUT	CX timeout in VBlank ticks; AX result
05h	COMM_SEND_STRING	DS:DX NUL-terminated; AX status
06h	COMM_SEND_BLOCK	DS:DX data, CX bytes; AX status
07h	COMM_RECEIVE_BLOCK	DS:DX buffer, CX capacity; AX status and returned byte count
08h	COMM_SET_TIMEOUT	BX receive, CX transmit; 0 immediate, FFFFh forever; default forever
09h/0Ah	COMM_SET/GET_BAUDRATE	0 = 9600, 1 = 38400
0Bh/0Ch	COMM_SET/GET_CANCEL_KEY	key mask; 0 disables; setter returns old mask

AH	Service	Register contract / notes
0Dh	COMM_XMODEM	DS:BX private state; reserved for system use and not guaranteed for applications

3.6 Sound - INT 15h (printed pages 37-41)

Four channels use 32 four-bit PCM samples per repeating waveform, 16 left and right volume levels, and an 11-bit pitch value. Channel 2 has 8-bit PCM voice, channel 3 one-shot frequency sweep, and channel 4 polynomial noise. FreyaBIOS 1.0 does not implement PCM-voice volume helpers; the C library supplies them.

AH	Service	Register contract / notes
00h	SOUND_INIT	disables all sound and clears wave table
01h/02h	SOUND_SET/GET_CHANNEL	BL flags: bits 0-3 channel enables, bit 5 voice, bit 6 sweep, bit 7 noise; bit 4 must be zero
03h/04h	SOUND_SET/GET_OUTPUT	BL input / AX output: speaker enable and output range, headphones enable, headphone-present read bit
05h	SOUND_SET_WAVE	AL channel 0-3, DS:DX 16 bytes containing 32 signed four-bit samples
06h/07h	SOUND_SET/GET_PITCH	AL channel, BX input / AX output 0-2047
08h/09h	SOUND_SET/GET_VOLUME	AL channel, BL input / AL output; low nibble right, high nibble left
0Ah/0Bh	SOUND_SET/GET_SWEEP	BL signed delta -128..127, CL step 0..31; step duration about 2.667 * (n + 1) ms
0Ch/0Dh	SOUND_SET/GET_NOISE	bits 0-2 shape, bit 3 reset, bit 4 enable; set bits 3 and 4 when starting noise
0Eh	SOUND_GET_RANDOM	AX polynomial-counter value 0-32767; channel 4 must be enabled

Pitch is 3,072,000 / ((2048 - n) * 32) Hz.

3.7 Calendar and timers - INT 16h (printed pages 42-44)

The cartridge contains an RTC; the console provides HBlank and VBlank timers. RTC fields are year (00 = 2000), month, date, day of week, hour, minute, second. The seven-byte datetime_t stores them in that order.

AH	Service	Register contract / notes
00h	RTC_RESET	factory/initialization use only; applications must not call it
01h/02h	RTC_GET/SET_DATETIME in the manual	BX field, AX result or CX input
03h/04h	RTC_SET/GET_DATETIME_STRUCT	DS:DX seven-byte structure
05h	RTC_ENABLE_ALARM	BL hour, BH minute; alarm arrives through cartridge IRQ hook
06h	RTC_DISABLE_ALARM	disables alarm
07h	TIMER_ENABLE	AL HBlank=0/VBlank=1, BL one-shot=0/auto=1, CX preset
08h	TIMER_DISABLE	AL timer; FreyaBIOS 1.0 bug reportedly changes HBlank period instead of disabling
09h	TIMER_GET_COUNT	AL timer; AX current count

HBlank timer frequency is approximately 12.00 kHz (83.33 us); VBlank is 75.47 Hz (13.25 ms). The manual and supplied TIMER.INC name AH 01h get and AH 02h set, while current libww and AthenaBIOS implement AH 01h set and AH 02h get. Because both halves of the open stack agree, this translation records the discrepancy without changing the SDK.

3.8 System - INT 17h (printed pages 45-52)

SYS_INTERRUPT_SET_HOOK takes AL source ID, DS:BX new intvector_t, and optional DS:DX old-vector output. The eight-byte vector is offset, code segment, DS to load for the handler, and a BIOS-owned reserved word. A zero code segment means the caller's segment. SYS_INTERRUPT_RESET_HOOK restores the saved vector; FreyaBIOS 1.0 may reset unrelated sources, so use the libww replacement. The cartridge/RTC IRQ can conflict with a user hook, and the manual reports the VBlank timer-count source broken in FreyaBIOS 1.0, possibly by stack corruption.

AH	Service	Register contract / notes
00h/01h	SYS_INTERRUPT_SET/RESET_HOOK	hook or restore a far handler
02h	SYS_WAIT	CX VBlank ticks, about 13.25 ms each
03h	SYS_GET_TICK_COUNT	DX:AX 32-bit VBlank ticks since boot
04h	SYS_SLEEP	panel off and sleep until configured wake key
05h/06h	SYS_SET/GET_SLEEP_TIME	idle minutes; zero disables automatic sleep
07h/08h	SYS_SET/GET_AWAKE_KEY	key mask; zero means any key
09h	SYS_SET_KEEPALIVE_INT	low eight bits select IRQs active during sleep; default key only
0Ah	SYS_GET_OWNERINFO	CX bytes to DS:DX; AX success/timeout; manual warns original call is wrong on Color and recommends wwc_sys_get_ownerinfo()
0Bh/0Ch	SYS_SUSPEND/RESUME	system-only core-image state save/restore
0Dh/0Eh	SYS_SET/GET_REMOTE	system-only remote-mode flag
0Fh	SYS_ALLOC_IRAM	BX optional address slot or zero for BIOS-managed handle, CX size; AX IRAM offset or zero
10h	SYS_FREE_IRAM	BX address; must free in reverse allocation order
11h	SYS_GET_MY_IRAM	AX allocation associated with caller's CS handle
12h	SYS_GET_VERSION	AX: major bits 12-15, minor 8-11, subminor 0-7
13h	SYS_SWAP	suspend current and resume selected user core image; system-only
14h/15h	SYS_SET/GET_RESUME	unpublished resume-save state; system-only

Suspend saves all 16 KiB IRAM, BIOS work, stack, VRAM, selected I/O state, SS, and SP into FreyaOS SRAM work. For automatic IRAM handles, CS identifies the owner and BIOS remembers at most eight entries.

Owner information is 16-byte name, 16-bit birth year, birth month/day, sex (0 unknown, 1 male, 2 female), and blood type (0 unknown, 1 A, 2 B, 3 O, 4 AB).

The stock SDK declares sys_get_tick_count() as 16-bit even though this table and AthenaBIOS return DX:AX. The WWTM overlay corrects it to uint32_t.

3.9 Bank services - INT 18h (printed pages 53-56)

Mapped regions are SRAM segment 1000h, ROM0 2000h, and ROM1 3000h. BANK_SET_MAP takes BL region ID 0/1/2 and CX bank number; SRAM accepts 0-3 and ROM 0-7. BANK_GET_MAP returns the bank in AX.

Physical access services use BX bank ID: SRAM 0000h-0003h, flash 8000h-8007h; DX is the offset.

AH	Service	Register contract / notes
00h	BANK_SET_MAP	BL region, CX bank
01h	BANK_GET_MAP	BL region, AX bank

AH	Service	Register contract / notes
02h	BANK_READ_BYTE	BX bank, DX offset, AL byte
03h	BANK_WRITE_BYTE	BX bank, DX offset, byte input
04h	BANK_READ_WORD	BX bank, DX offset, AX 16-bit word
05h	BANK_WRITE_WORD	BX bank, DX offset, CX word
06h	BANK_READ_BLOCK	BX bank, DX offset, CX length, DS:SI output
07h	BANK_WRITE_BLOCK	BX bank, DX offset, CX length, DS:SI input
08h	BANK_FILL_BLOCK	BX bank, DX offset, CX length (0 means 64 KiB), AL fill byte
09h	BANK_ERASE_FLASH	system-only, details unpublished

The manual reverses "read" and "write" in the prose under the two block calls; their names, buffers, and current implementations make the intended direction clear. The stock SDK's bank setter macros call the getter and its `bank_read_word()` return type is only eight bits. Both are corrected by the overlay.

4. FreyaOS (printed pages 57-77)

4.1 OS layout, boot, and startup

FreyaOS supplies processes, filesystems, indirect libraries (ILs), and C libraries. The original vendor supported its supplied C toolchain, not arbitrary environments. FreyaOS owns all memory:

- flash banks 0-5: 384 KiB FreyaOS filesystem;
- flash bank 6: 64 KiB FreyaOS;
- flash bank 7: 64 KiB FreyaBIOS;
- SRAM bank 0: 64 KiB soft filesystem;
- SRAM banks 1-2: two 64 KiB user processes; and
- SRAM bank 3: 64 KiB OS work/process area.

Programs link code for physical `20000h` and data for `10000h`, select one of the display-specific startup objects, and place `ProcContext` at the start of `DGROUP`/SRAM. ILs use a different startup contract.

At boot BIOS invokes FreyaOS. FreyaOS checks cartridge consistency and, if uninitialized or damaged, runs self-test and initializes both flash and SRAM filesystems, erasing everything except BIOS and OS. It reads `/rom0/init.rc`, loads configured RunnableILs, and selects a shell. Holding Y1+Y4 selects simple mode with current directory `/kern` and built-in Meg shell. Otherwise normal mode starts in `/rom0`, running `/rom0/@shell` if present or Meg otherwise.

`init.rc` syntax:

- `$var=value` defines a system-wide environment variable, up to 64 bytes, retrievable with `getenv()`.
- `@ilib [arguments...]` searches `/rom0` then `/kern` and launches an IL as a RunnableIL. On failure it displays a message and waits for A before continuing.

A user shell is a RunnableIL stored as `/rom0/@shell`. Simple mode bypasses a broken `init.rc` or user shell.

Normal startup has two entries. The load entry copies `DGROUP` initialized data to SRAM and far-returns. The run entry sets DS and ES to `1000h`, clears scroll, sets full-screen windows, selects Screen 2 text, chooses the startup text mode, sets unpublished VRAM registers, hides sprites, enables the appropriate screens, calls `_main`, then calls `INT 10h`.

4.2 Processes

FreyaOS itself and applications are processes with independent storage and execution state. Suspend/resume permits a child to run without destroying its parent and can preserve a state across power-off. Three PCBs exist: one for the OS and two for users. Status values are 0 unused, 1 loading, 2 running, and 3 suspended. Exit status 0 means success; -1 means load failure.

Launch sequence: allocate a free PCB; call the executable's `_load0` to copy initialized `DGROUP` into its 64 KiB SRAM bank and return a far `_run0` pointer; inherit parent IL pointers and working directory; copy argv into remaining data space; set `_heap` to the remainder; call `_run0`, which configures display and stack, calls `main`, and exits through `INT 10h`.

`ProcContext` begins at `1000h:0000h` and contains:

Field	Meaning
`_id[4]`	compiler tag such as `LCC\0` or `TCC\0`
`_pid`, `_ppid`	process serial ID and parent ID; OS PID 0, OS parent -1
`_pcbid`, `_ppcbid`	internal PCB IDs
`_ilib`, `_proc`	inherited far pointers to IlibIL and ProcIL
`_cwfs`, `_currentdir[64]`	working filesystem and directory
`_argv`	near argv array
`_resource`	far resource pointer
`_heap`	near start of free heap

4.3 Filesystem

FreyaOS 1.3 has fixed top-level directories and no user-created directories, though the format leaves room for them. Storage is allocated in 128-byte blocks. XMODEM transfers files. Only `/rom0` can be directly mapped with `mmap()`.

Directory	Storage	Capacity and role
`/ram0`	64 KiB SRAM	up to 64 files; saves and temporary data; programs cannot execute here
`/rom0`	384 KiB flash	up to 128 programs/ILs; files may be mapped directly
`/kern`	64 KiB OS ROM	built-in Meg and stream drivers; user cannot add files

Each file has a 64-byte entry: 16-byte name, 24-byte description, far location, 32-bit byte length, 16-bit 128-byte block count, mode, packed modification time, future application ID, and resource pointer. Mode bits are `0001h` executable, `0002h` writable, `0004h` readable, `0008h` forbid `mmap`, `0010h` stream IL, `0020h` IL, `0040h` future symlink, and `0080h` future directory. IL filenames begin with `@`.

Packed modification time: year since 2000 in bits 25-31, month 20-24, day 15-19, hour 11-14, minute 5-10, and seconds divided by two in bits 0-4.

4.4 Indirect libraries

An IL is dynamically linked at runtime and exposes a structure of far function pointers. Swapping an IL can extend behavior; shared libraries are loaded once, and large programs can be split into small modules. FreyaOS 1.3 includes IlibIL, ProcIL, FsIL, StreamIL, and RunnableIL.

IlibIL resolves a named IL into a caller-provided structure. `_open` searches `/rom0` then `/kern`; `_open_system` searches only `/kern`. Placing `/rom0/@ilib` replaces the system resolver at boot.

ProcIL supplies `_load`, `_run`, `_exec`, `_exit`, `_yield`, `_suspend`, `_resume`, and `_swap`. `_exec` combines load and run. `_exit` terminates through `INT 10h`. Suspend and swap operate on PCB IDs. `/rom0/@proc` replaces the system process IL.

FsIL supplies entry enumeration/search, `mmap`, open/close/read/write/lseek, chmod, freeze/melt, create/unlink, initialize/defragment, and free-space queries. It is intended for the C file API rather than direct application calls. StreamIL supplies open/close/read/write for stream-backed files and devices. RunnableIL has a process-like `_exec(argc, argv)` and is used for shells.

A user IL must begin with its IL method structure, supply an `IL` prefix and `ILinfo`, receive C arguments on the stack, preserve all registers except return registers, omit ordinary program startup, and far-return. DS remains the caller's DS; an IL that needs its own `DGROUP` uses `c0ilib.obj` and `il_get_ds()`. Unresolved far method segments may be zero because IlibIL fixes them when the file is opened. Every IL must provide far `_get_info()` returning `ILinfo`.

The common `IL` prefix is far `link_pos`, method count (including `_get_info`), and far `_get_info` pointer. `ILinfo` contains far strings for class, name, version, description, and dependencies; this manual says only name and version were meaningfully used.

Original compiler ABI: `char`, `int`, and near pointer arguments occupy one 16-bit stack word; `long` and far pointer arguments occupy two. Return `char` in AX with AH zero, `int`/near pointer in AX, and `long`/far pointer in DX:AX.

4.5 C and assembly library ABI

FreyaOS functions are supplied in Microsoft-compatible `.lib` archives. C symbols have a leading underscore. Push arguments right-to-left; the caller removes them. `char` occupies two stack bytes with high byte zero, `int` two, and double/far-sized values four, with the documented word order. Returns use AX for char/int and DX:AX for four-byte values. The small model uses near calls; library calls may destroy AX, BX, and DX. The main archives are `libww.lib` and `libwwc.lib`.

5. Supplied material (printed page 78)

`include.asm/`, written by Watanabe, contains assembly constants and macros for FreyaBIOS and `libwwc.lib`. The author asks that only readers who understand the definitions use them. `sample/il_test/` is an assembly user IL called from C and is intended as a reference for building user ILs.

The supplied include files also contain typographical bugs. Examples include `puah`, `_wwc_get_color_mede`, calls to the palette setter from the palette getter, `cahr_count`, and an `end` where a macro needs `endm`. They are preserved in the UTF-8 source conversion; they should be treated as research material, not drop-in production headers.

6. Afterword and reliability warning (printed page 79)

The author states that some sections had not yet been checked by writing and running programs, so incorrect explanations may remain. Readers were invited to report mistakes, missing explanations, and unclear expressions. This warning is important when reconciling the manual with executable firmware.

7. Author and revision note (printed page 80)

Atsushi Watanabe, born February 4, 1979, graduated from Tokyo Metropolitan College of Technology's mechanical engineering department in 1999, joined a semiconductor manufacturer, and joined Digitalis in 2000. Listed works include Tako Project, FM Sound Player, and WonderSwan Total Sound Driver. This copy is marked an interim publication dated January 24, 2002.

Confirmed SDK consequences

The manual was compared with the installed Wonderful `libww` headers and pinned AthenaBIOS source. Four defects were independently corroborated and fixed by the repository overlay:

1. All three `*_set_map()` convenience macros call `bank_get_map()` with two arguments; they must call `bank_set_map()`.
2. `bank_read_word()` is declared `uint8_t` but AH `04h` returns a word in AX; AthenaBIOS loads AX.
3. `sys_get_tick_count()` is declared `uint16_t` but AH `03h` returns DX:AX; AthenaBIOS maintains and returns both words.
4. `LCD_SLEEP_ON`/`OFF` values are reversed. AH `1Fh` and the hardware control bit use 0 for display enabled and 1 for disabled/sleeping.

See `../../sdk/wonderwitch-wwtm/README.md` and run `python3 ../../scripts/wonderwitch_sdk_contract_test.py` from this directory.