

Started: September 25, 2004 / Current version: January 16, 2005

# Unofficial Pokémon-Mini™ Hardware Document

**Sources:** JustBurn (entire document)  
p0p (spelling check & correction)  
Dave/X (instruction codes & some Registers)  
Asterick (RTC/EEPROM access)  
DarkFader (cart pins-out)

---

## *Contents of the document:*

- . Pokémon-Mini specifications
- . Hardware memory-registers (special registers I/O)
- . Video & audio processing
- . Instruction codes of the CPU & Timing
- . Accessing internal EEPROM/RTC
- . Hardware cart-slot pin-out

**NOTE:** This document is for homebrew development only and will not work for Commercial game emulation.  
(Lines marked **RED** aren't available into this version of the document)

---

*Version 0.90 by JustBurn*

## **0. Table of contents**

- 1. Introduction**
- 2. Pokémon-Mini specifications**
- 3. Memory map**
- 4. Game header**
- 5. Special registers I/O**
- 6. Video processing**
- 7. Audio processing**
- 8. CPU information**
- 9. CPU instructions**
- 10. Access external RTC / EEPROM**
- 11. Hardware cart-slot pin-out**

## 1. Introduction

PM (Pokémon-Mini) is one of the smallest and most amazing handheld machines created by Nintendo, it allows playing games inside a very tinny cartridge slot, but that is not all! Pokémon-Mini features a Rumble Motor, Shock Detector and 6-slots for savegames build-in into a very small system, it uses up to 8MegaBits cartridge, has 1-Channel Sound and it displays background-tiles and sprites too!



## 2. Pokémon-Mini specifications

**CPU:** *8-Bits Nintendo custom IC, clocked 4.00 MHz*

**BIOS:** *4096 Bytes, read-only*

**RAM:** *4096 Bytes:  
First 1056 bytes reserved for video  
Last 3040 bytes for general purpose*

**ROM:** *Up to 1 Megabyte (8 Megabits)  
Code is banked at 32KBytes*

**VIDEO:** *96x64 Mono-LCD  
Display tiles (with 3 different map sizes)  
Display sprites (16x16, up to 24 sprites)  
Flickering allows to add a 3<sup>rd</sup> color  
Contrast Control  
Moving tile-map (not scrolling!!!)*

**SOUND:** *3 Levels of volume  
Single-channel square-wave with adjustable pulse-width  
Can read back timer values (since sound is assigned  
to Timer 3)*

**TIMERS:** *24-Bits seconds-timer (increments each second)  
8-Bits 256Hz timer (increments each 1/256 of second)  
3x 16-Bits timers with configurable pre-scale  
(Timer 3 configured for sound output)*

**EXTRAS:** *Rumble motor  
Shock detector  
6-Slots RAM for savegames  
RTC (real time clock) Build-in  
IR transmitter / receiver*

**POWER:** *1.5V (converted to 3.3V internally) with 1 AAA Battery*

**VIDEO TIMING:** *Refresh Rate: Variable (default: 30Hz)  
Horizontal sync: unknown  
Vertical sync: unknown*

### 3. Memory map

Address Range	Description
0x0000-0x0FFF	Internal BIOS
0x1000-0x12FF	Video buffer (doesn't write if render is on!)
0x1300-0x135F	Sprites attrib. memory (OAM)
0x1360-0x141F	Background tile-mapping
0x1420-0x1FFF	General purpose RAM
0x2000-0x20FF	Special registers I/O (for hardware control)
0x2100-0x21CF	Game header
0x21D0-0x7FFF	Game fixed-bank 0
0x8000-0xFFFF	Game selectable-bank 0...31

Code is accessed in 16-Bits range where:

0x2100 – 0x7FFF = fixed Bank 0

0x8000 – 0xFFFF = selectable bank 0...31

Data is accessed in full memory range (20-Bits) and isn't affected by bank selection (0x2100 – 0x7FFF always points to 0x02100 – 0x07FFF)

#### **Internal BIOS: 0x0000 – 0x0FFF (4096 Bytes)**

*This area is accessed when Pokémon-mini is powered on, CPU will start at 0x009A by initializing hardware, video, CPU, check cartridge, reports battery status (low or normal) and much more.*

*Any interrupt called will pass by BIOS first, also when using CINT/JINT instructions, it will jump to a table at the start of the BIOS and execute the wanted command. (read-only)*

#### **Video buffer: 0x1000 – 0x12FF (786 Bytes)**

*This area is reserved for background/sprites rendering (inside hardware), when render is active, any access to this area will be nulled (rendering will overwrite the value), when render is deactivated, you can freely write to this area and will affect the display directly.*

#### **Sprites attrib. memory: 0x1300 – 0x135F (96 Bytes)**

*Read & write area that is reserved for sprites information.*

*Each sprite contains 4 Bytes of data that is used to plot a single 16x16 Sprite (with mask) onto screen.*

#### **Background tile-mapping: 0x1360 – 0x141F (192 Bytes)**

*Each byte in this area points to a tile (8x8 pixels) that should be displayed on the screen, up to 256 totally different tiles can be shown at same time by writing 0x00(0) to 0xFF(255).*

*The mapping depends of what was chosen in REGISTER 0x80.*

**General purpose RAM: 0x1420 – 0x1FFF (3040 Bytes)**

*The user is free to read and write anything into this area, but keep in mind that the high end of this RAM is reserved for stacking.*

**Special registers IO: 0x2000 – 0x20FF (256 Bytes)**

*Any write access in this area will affect the hardware directly; there are special cases that some bits cannot be written or read.*

**Game header: 0x2100 – 0x21CF (208 Bytes)**

*Identifies the cartridge and informs what commands the events should execute, please refer to "4. Game header" for more information. (read-only)*

**Game Fixed-Bank 0: 0x21D0 – 0x7FFF (24112 Bytes)**

*First bank in cartridge where important code & data are stored and require to be accessed all the time. (read-only)*

**Game Selectable-Bank: 0x8000 – 0xFFFF (32768 Bytes)**

*Selectable bank (CPU Register "V" points to the bank) that holds information (code or data) which isn't required to be always present, in code: up to 32 Banks can be selected and accessed (8Megabits card); in data: all the content can be accessed ignoring the bank location 0x00000 – 0xFFFFF (8Megabits card). (read-only)*

## 4. Game header

Address Range	Description
0x02100-0x02101	Cartridge ID
0x02102-0x02107	Event #0 code – Game start-up
0x02108-0x0210D	Event #1 code – V-Blank interrupt
0x0210E-0x02113	Event #2 code – V-Draw interrupt
0x02114-0x02119	Event #3 code – Timer 2 overflow
0x0211A-0x0211F	Event #4 code – Unknown
0x02120-0x02125	Event #5 code – Timer 1 overflow
0x02126-0x0212B	Event #6 code – Unknown
0x0212C-0x02131	Event #7 code – Timer 3 overflow
0x02132-0x02137	Event #8 code – Timer 3 overflow
0x02138-0x0213D	Event #9 code – Unknown
0x0213E-0x02143	Event #10 code – Unknown
0x02144-0x02149	Event #11 code – Unknown
0x0214A-0x0214F	Event #12 code – Unknown
0x02150-0x02155	Event #13 code – IR receive low to high
0x02156-0x0215B	Event #14 code – Shock detector
0x0215C-0x02161	Event #15 code – Key press: Power button
0x02162-0x02167	Event #16 code – Key press: D-pad right
0x02168-0x0216D	Event #17 code – Key press: D-pad left
0x0216E-0x02173	Event #18 code – Key press: D-pad down
0x02174-0x02179	Event #19 code – Key press: D-pad up
0x0217A-0x0217F	Event #20 code – Key press: “C” key
0x02180-0x02185	Event #21 code – Key press: “B” key
0x02186-0x0218B	Event #22 code – Key press: “A” key
0x0218C-0x02191	Event #23 code – Unknown
0x02192-0x02197	Event #24 code – Unknown
0x02198-0x0219D	Event #25 code – Unknown
0x0219E-0x021A3	Reserved – must be 0
0x021A4-0x021AB	“NINTENDO” string
0x021AC-0x021AF	4 Bytes – Game code (customized by user)
0x021B0-0x021BB	12 Bytes – Game name (customized by user)
0x021BC-0x021BD	Contain string “2P” (2 players!?)
0x021BE-0x021CF	Reserved – must be 0
0x021D0-0xFFFF	...game area...

### **Cartridge ID: 0x2100 – 0x2101**

*On Nintendo's development cart, it identifies the type of flash IC used. On commercial games (and homebrew games), it identifies that it's a valid cartridge... value must be 0x4D 0x4E (string "MN") or Pokémon-Mini will not detect the cart (Invalid header).*

### **Event #0 – Game start-up: 0x2102 – 0x2107**

*Code executed when the BIOS finished it's initializations, most games change U(V after jump) to point Bank 0 or 1 and then Jump, this event never returns.*

### **Event #1 – V-Blank interrupt: 0x2108 – 0x210D**

*Each time the PM triggers V-Sync to re-init vertical position of LCD, it generates Interrupt #1, If interrupt enabled: PC automatically jumps here and executes the desired code ending with "RETI" instruction. Cleaning interrupt flag is required! REGISTER 0x27 – Bit 7 (0x80)*

### **Event #2 – V-Draw interrupt: 0x210E – 0x2113**

*Same as event #1 but it's triggered at the end of V-Sync. Interrupt #2 Cleaning interrupt flag is required! REGISTER 0x27 – Bit 6 (0x40)*

### **Event #3 – Timer 2 overflow: 0x2114 – 0x2119**

*When timer 2 counter overflow (0xFFFF to 0x0000), it generates Interrupt #3, If interrupt enabled: PC automatically jumps here and executes the desired code ending with "RETI" instruction. Cleaning interrupt flag is required! REGISTER 0x27 – Bit 5 (0x20)*

### **Event #4 – Unknown: 0x211A – 0x211F**

*Conditions to trigger this event are still unknown. Cleaning interrupt flag is required! REGISTER 0x27 – Bit 4 (0x10)*

### **Event #5 – Timer 1 overflow: 0x2120 – 0x2125**

*When timer 1 counter overflow (0xFFFF to 0x0000), it generates Interrupt #5, If interrupt enabled: PC automatically jumps here and executes the desired code ending with "RETI" instruction. Cleaning interrupt flag is required! REGISTER 0x27 – Bit 3 (0x08)*

### **Event #6 – Unknown: 0x2126 – 0x212B**

*Conditions to trigger this Event are still unknown. Cleaning interrupt flag is required! REGISTER 0x27 – Bit 2 (0x04)*

### **Event #7 – Timer 3 overflow: 0x212C – 0x2131**

*When timer 3 counter overflow (0xFFFF to 0x0000), it generates Interrupt #7, If interrupt enabled: PC automatically jumps here and executes the desired code ending with "RETI" instruction. Cleaning interrupt flag is required! REGISTER 0x27 – Bit 1 (0x02)*

### **Event #8 – Timer 3 overflow: 0x2132 – 0x2137**

*Clone of event #7. Interrupt #8 (This cloned event have something to do with timer being reserved for Sound!?) Cleaning interrupt flag is required! REGISTER 0x27 – Bit 0 (0x01)*



### **Event #9 to #12 – Unknown: 0x2138 – 0x2155**

*Conditions to trigger these events are still unknown.*

*Cleaning interrupt flag is required!*

*Event #9 -> REGISTER 0x28 – Bit 5 (0x20)*

*Event #10 -> REGISTER 0x28 – Bit 4 (0x10)*

*Event #11 -> REGISTER 0x28 – Bit 3 (0x08)*

*Event #12 -> REGISTER 0x28 – Bit 2 (0x04)*

### **Event #13 – IR receive low to high: 0x2156 – 0x215B**

*When a IR device emit light low to high (Off to On), it generates interrupt #14, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x2A – Bit 7 (0x80)*

### **Event #14 – Shock detector: 0x2156 – 0x215B**

*If the user shakes the Pokémon-Mini device, it generates Interrupt #14, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x2A – Bit 6 (0x40)*

### **Event #15 – Key press: Power button: 0x215C – 0x2161**

*When user presses (release isn’t triggered) the “power” key, it generates Interrupt #15, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x29 – Bit 7 (0x80)*

### **Event #16 – Key press: DPAD right: 0x2162 – 0x2167**

*When user presses (release isn’t triggered) the “right” on D-pad, it generates Interrupt #16, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x29 – Bit 6 (0x40)*

### **Event #17 – Key press: DPAD left: 0x2168 – 0x216D**

*When user presses (release isn’t triggered) the “left” key, it generates Interrupt #17, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x29 – Bit 5 (0x20)*

### **Event #18 – Key press: DPAD down: 0x216E – 0x2173**

*When user presses (release isn’t triggered) the “down” key, it generates Interrupt #18, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x29 – Bit 4 (0x10)*

### **Event #19 – Key press: DPAD up: 0x2174 – 0x2179**

*When user presses (release isn’t triggered) the “up” key, it generates Interrupt #19, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x29 – Bit 3 (0x08)*

**Event #20 – Key press: “C” key: 0x217A – 0x217F**

*When user presses (release isn’t triggered) the “C” key, it generates Interrupt #20, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x29 – Bit 2 (0x04)*

**Event #21 – Key press: “B” key: 0x2180 – 0x2185**

*When user presses (release isn’t triggered) the “B” key, it generates Interrupt #21, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x29 – Bit 1 (0x02)*

**Event #22 – Key press: “A” key: 0x2186 – 0x218B**

*When user presses (release isn’t triggered) the “A” key, it generates Interrupt #22, If interrupt enabled: PC automatically jumps here and executes the desired code ending with “RETI” instruction.*

*Cleaning interrupt flag is required! REGISTER 0x29 – Bit 0 (0x01)*

**Event #23 to #25 – Unknown: 0x218C – 0x219D**

*Conditions to trigger these Events are still unknown.*

**“NINTENDO” word: 0x21A4 – 0x21AB**

*Content of this area MUST BE “NINTENDO” string (case-sensitive):*

*0x4E 0x49 0x4E 0x54 0x45 0x4E 0x44 0x4F.*

*If content isn’t valid... Pokémon-Mini will not detect the cart (invalid header).*

**4 Bytes – Game Code: 0x21AC – 0x21AF**

*The user is free to put anything in this area but must be a string (ASCII), examples of some codes from original cards:*

*“MPTP” – Pokémon-Mini Party Europe*

*“MPBE” – Pokémon-Mini Pinball Europe*

**12 Bytes – Game Name: 0x21B0 – 0x21BB**

*The user is free to put anything in this area but must be a string (ASCII), examples of some names from original cards:*

*“Party Mini E” – Pokémon-Mini Party Europe*

*“Pinball” – Pokémon-Mini Pinball Europe*

*Unused space is filled with 0x00.*

**Cartridge ID: 0x21BC – 0x21BD**

*All commercial cards have 0x32 0x50 (string “2P”) in this area (Refers to 2 Players!?!?), changing any byte has no effect on the BIOS.*

## 5. Special registers I/O

Address	Description
0x2000	Start-up contrast
0x2001	CPU related!?
0x2002	CPU related!?
0x2003-0x2007	Unused
0x2008	Seconds-timer control
0x2009-0x200B	Seconds-timer counter
0x200C-0x200F	Unused
0x2010	Low power detector
0x2011-0x2017	Unused
0x2018	Timer 1 pre-scale + enable
0x2019	Timers 1 to 3 enabler
0x201A	Timer 2 pre-scale + enable
0x201B	Unknown
0x201C	Timer 3 pre-scale + enable
0x201D	Unknown
0x201E-0x201F	Unused
0x2020	Interrupt #1-#8 primary enable
0x2021	Interrupt #15-#22 primary enable
0x2022	Interrupt #13-#14 primary enable
0x2023	Interrupt #1-#8 secondary enable
0x2024	Interrupt #9-#12 secondary enable
0x2025	Interrupt #15-#22 secondary enable
0x2026	Interrupt #13-#14 secondary enable
0x2027	Interrupt flag #1-#8
0x2028	Interrupt flag #9-#12
0x2029	Interrupt flag #15-#22
0x202A	Interrupt flag #13-#14
0x202B-0x202F	Unused
0x2030	Timer 1 control 1
0x2031	Timer 1 control 2
0x2032-0x2033	Timer 1 preset value
0x2034-0x2035	Timer 1 sound-pivot (Unused)
0x2036-0x2037	Timer 1 counter

Address	Description
0x2038	Timer 2 control 1
0x2039	Timer 2 control 2
0x203A-0x203B	Timer 2 preset value
0x203C-0x203D	Timer 2 sound-pivot (Unused)
0x203E-0x203F	Timer 2 counter
0x2040	256 Hz timer control
0x2041	256 Hz timer counter
0x2042-0x2043	Unused
0x2044	Unknown
0x2044	Unknown
0x2045	Unknown
0x2046	Unknown
0x2047	Unknown
0x2048	Timer 3 control 1
0x2049	Timer 3 control 2
0x204A-0x204B	Timer 3 preset value
0x204C-0x204D	Timer 3 sound-pivot
0x204E-0x204F	Timer 3 counter
0x2050	Unknown
0x2051	Unknown
0x2052	Keypad status
0x2053	Unknown
0x2054	Unknown
0x2055	Unknown
0x2056-0x205F	Unused
0x2060	I/O peripheral circuit select
0x2061	I/O peripheral status/control
0x2062	Unknown
0x2063-0x206F	Unused
0x2070	Unknown
0x2071	Sound volume
0x2072-0x207F	Unused

Address	Description
0x2080	LCD control
0x2081	LCD render refresh-rate
0x2082-0x2084	BG tile data memory offset
0x2085	BG vertical move
0x2086	BG horizontal move
0x2087-0x2089	Sprite tile data memory offset
0x208A	LCD status
0x208B-0x208F	Unused
0x2090-0x20EF	Unused
0x20F0-0x20F7	Unused
0x20F8-0x20FD	Unused
0x20FE	Direct-LCD control / data
0x20FF	Direct-LCD data

## **REGISTER 0x00 – Start-up contrast**

**Start-up value: 0x83** (default)

**Description:**

*Seems to control the contrast of LCD when the user turns-on the Pokémon-Mini console, changing the lower 2bits will freeze the console.*

Bits	R/W	Description
Bit 0 – 1	Read/Write	THEY MUST BE 1
Bit 2 – 7	Read/Write	Start-Up Contrast (doesn't affect contrast until user reboots his PM)

## **REGISTER 0x01 – CPU related!?**

**Start-up value: 0x20 or 0x30**

**Description:**

*This register seems to be related to CPU.*

Bits	R/W	Description
Bit 0 – 7	Unknown	Unknown

## **REGISTER 0x02 – CPU related!?**

**Start-up value: 0x5C or 0x5E**

**Description:**

*This register seems to be related to CPU.*

Bits	R/W	Description
Bit 0 – 7	Unknown	Unknown

## **REGISTER 0x08 – Seconds-timer control**

**Start-up value: 0x00 or 0x01**

**Description:**

*Controls seconds-timer.*

Bits	R/W	Description
Bit 0	Read/Write	0=Timer paused / 1=Timer running
Bit 1	Write Only	1=Timer reset (counter becomes 0x000000)
Bit 2 – 7	Unused	Unused

### **REGISTER 0x09 – Seconds-Timer counter (part 1)**

**Start-up value:** 0x??

**Description:**

*Value of 8 lower-bits of seconds-timer.*

Bits	R/W	Description
Bit 0 – 7	Read Only	Low byte of 24-bits seconds-timer counter

### **REGISTER 0x0A – Seconds-timer counter (part 2)**

**Start-up value:** 0x??

**Description:**

*Value of 8 middle-bits of seconds-timer.*

Bits	R/W	Description
Bit 0 – 7	Read Only	Med. byte of 24-bits seconds-timer counter

### **REGISTER 0x0B – Seconds-timer counter (part 3)**

**Start-up value:** 0x??

**Description:**

*Value of 8 higher-bits of seconds-timer.*

Bits	R/W	Description
Bit 0 – 7	Read Only	High byte of 24-bits seconds-timer counter

### **REGISTER 0x10 – Low power detector**

**Start-up value:** 0x??

**Description:**

*Report battery status and is related with something else.*

Bits	R/W	Description
Bit 0 – 4	Read/Write	Unknown
Bit 5	Read Only	0=Battery OK / 1=Battery weak
Bit 6 – 7	Unused	Unused

### **REGISTER 0x18 – Timer 1 pre-scale + enable**

**Start-up value: 0xFF**

**Description:**

*Sets timer 1 pre-scale and enables counting.*

Bits	R/W	Description
Bit 0 – 2	Read/Write	Pre-scale: 0: 2000000 Hz 1: 500000 Hz 2: 125000 Hz 3: 62500 Hz 4: 31250 Hz 5: 15625 Hz 6: 3906.25 Hz 7: 976.5625 Hz
Bit 3	Read/Write	Enable counting
Bit 4 – 7	Read/Write	Unused

### **REGISTER 0x19 – Timers 1 to 3 enabler**

**Start-up value: 0x03**

**Description:**

*Enables timers 1 to 3 circuits.*

Bits	R/W	Description
Bit 0	Read/Write	SET IT TO "0"
Bit 1 – 3	Unused	Unused
Bit 4	Read/Write	SET IT TO "0"
Bit 5	Read/Write	Enable timers 1 to 3 circuits
Bit 6 – 7	Unused	Unused

### **REGISTER 0x1A – Timer 2 pre-scale + enable**

**Start-up value: 0xD0**

**Description:**

*Sets timer 2 pre-scale and enables counting.*

Bits	R/W	Description
Bit 0 – 2	Read/Write	Pre-scale: 0: 2000000 Hz 1: 500000 Hz 2: 125000 Hz 3: 62500 Hz 4: 31250 Hz 5: 15625 Hz 6: 3906.25 Hz 7: 976.5625 Hz
Bit 3	Read/Write	Enable counting
Bit 4 – 7	Read/Write	Unused



## **REGISTER 0x1C – Timer 3 pre-scale + enable**

**Start-up value: 0x08**

**Description:**

*Sets Timer 3 pre-Scale and enables counting.*

Bits	R/W	Description
Bit 0 – 2	Read/Write	Pre-Scale: 0: 2000000 Hz                      4: 31250 Hz 1: 500000 Hz                      5: 15625 Hz 2: 125000 Hz                      6: 3906.25 Hz 3: 62500 Hz                      7: 976.5625 Hz
Bit 3	Read/Write	Enable counting
Bit 4 – 7	Read/Write	Unused

## **REGISTER 0x20 – Event #1-#8 primary enable**

**Start-up value: 0x00**

**Description:**

*Activate events #1-#8 (primary).*

*For a event to be called successfully, primary (group of Interrupts), secondary (for each interrupt) and master interrupt (entire interrupts) must be set.*

Bits	R/W	Description
Bit 0 – 1	Read/Write	Timer 3 overflow (both bits must be set) Interrupt #7 – Interrupt 8
Bit 2 – 3	Read/Write	Timer 1 overflow (Both bits must be set) Interrupt #5 – Interrupts #6
Bit 4 – 5	Read/Write	Timer 2 overflow (Both bits must be set) Interrupt #3 – Interrupts #4
Bit 6 – 7	Read/Write	V-Draw/V-Blank trigger (Both bits must be set) Interrupt #1 – Interrupts #2

## **REGISTER 0x21 – Event #15-#22 primary enable**

**Start-up value: 0x30**

### **Description:**

*Activate events #15-#22 (primary).*

*For a event to be called successfully, primary (group of interrupts), secondary (for each interrupt) and master interrupt (entire interrupts) must be set.*

Bits	R/W	Description
Bit 0 – 1	Read/Write	Unknown
Bit 2 – 3	Read/Write	All keypad interrupts (Both bits must be set) Interrupt #15 – Interrupts #22
Bit 4 – 7	Read/Write	Unknown

## **REGISTER 0x22 – Event #9-#14 primary enable**

**Start-up value: 0x02**

### **Description:**

*Activate events #9-#14 (primary).*

*For a event to be called successfully, primary (group of interrupts), secondary (for each interrupt) and master interrupt (entire interrupts) must be set.*

Bits	R/W	Description
Bit 0 – 1	Read/Write	All #9 to #14 events (Both bits must be set) Interrupt #9 – Interrupts #14
Bit 2 – 7	Unused	Unused

## **REGISTER 0x23 – Event #1-#8 secondary enable**

**Start-up value: 0x00**

### **Description:**

*Activate events #1-#8 (secondary).*

*For a event to be called successfully, primary (group of interrupts), secondary (for each interrupt) and master interrupt (entire interrupts) must be set.*

Bits	R/W	Description
Bit 0	Read/Write	Timer 3 overflow (mirror) Enable Interrupt #8
Bit 1	Read/Write	Timer 3 overflow Enable Interrupt #7
Bit 2	Read/Write	Not called... Enable Interrupt #6
Bit 3	Read/Write	Timer 1 overflow Enable Interrupt #5
Bit 4	Read/Write	Not called... Enable Interrupt #4
Bit 5	Read/Write	Timer 2 overflow Enable Interrupt #3
Bit 6	Read/Write	V-Draw trigger Enable Interrupt #2
Bit 7	Read/Write	V-Blank trigger Enable Interrupt #1

## **REGISTER 0x24 – Event #9-#12 secondary enable**

**Start-up value: 0x02**

### **Description:**

*Activate events #9-#12 (secondary).*

*For a event to be called successfully, primary (group of interrupts), secondary (for each interrupt) and master interrupt (entire interrupts) must be set.*

Bits	R/W	Description
Bit 0 – 5	Read/Write	Unknown
Bit 6 – 7	Unused	Unused

## **REGISTER 0x25 – Event #15-#22 secondary enable**

**Start-up value: 0x00**

### **Description:**

*Activate events #15-#22 (secondary).*

*For a event to be called successfully, Primary (group of interrupts), Secondary (Each interrupt) and Master Interrupt (entire interrupts) must be set.*

Bits	R/W	Description
Bit 0	Read/Write	Press key "A" event Enable interrupt #22
Bit 1	Read/Write	Press key "B" event Enable interrupt #21
Bit 2	Read/Write	Press key "C" event Enable interrupt #20
Bit 3	Read/Write	Press D-pad up key event Enable interrupt #19
Bit 4	Read/Write	Press D-pad down key event Enable interrupt #18
Bit 5	Read/Write	Press D-pad left key event Enable interrupt #17
Bit 6	Read/Write	Press D-pad right key event Enable interrupt #16
Bit 7	Read/Write	Press power button event Enable interrupt #15

## **REGISTER 0x26 – Event #13-#14 secondary enable**

**Start-up value: 0x00**

### **Description:**

*Activate events #13-#14 (secondary).*

*For a event to be called successfully, primary (group of interrupts), secondary (each interrupt) and master interrupt (entire interrupts) must be set.*

Bits	R/W	Description
Bit 0 – 2	Read/Write	Unknown
Bit 3	Unused	Unused
Bit 4 – 5	Read/Write	Unknown
Bit 6	Read/Write	Shock detector trigger Enable interrupt #14
Bit 7	Read/Write	IR receiver - low to high trigger Enable interrupt #13

## **REGISTER 0x27 – Interrupt flag #1-#8**

**Start-up value: 0x00**

### **Description:**

*Interrupt flag of events #1-#8.*

*When flag is set, the interrupt is constantly called until it's cleared.*

*If "0" is written to a bit, interrupt flag is ignored*

*If "1" is written to a bit, interrupt flag is cleared*

Bits	R/W	Description
Bit 0	S-R flip-flop reset pin	Timer 3 overflow (mirror) Clear interrupt #8
Bit 1	S-R flip-flop reset pin	Timer 3 overflow Clear interrupt #7
Bit 2	S-R flip-flop reset pin	Not called... Clear interrupt #6
Bit 3	S-R flip-flop reset pin	Timer 1 overflow Clear interrupt #5
Bit 4	S-R flip-flop reset pin	Not called... Clear interrupt #4
Bit 5	S-R flip-flop reset pin	Timer 2 overflow Clear interrupt #3
Bit 6	S-R flip-flop reset pin	VDraw trigger Clear interrupt #2
Bit 7	S-R flip-flop reset pin	VBlank trigger Clear interrupt #1

## **REGISTER 0x28 – Interrupt flag #9-#12**

**Start-up value: 0x00**

### **Description:**

*Interrupt flag of events #9-#12.*

*When flag is set, the interrupt is constantly called until it's clear.*

*If "0" is written to a bit, interrupt flag is ignored*

*If "1" is written to a bit, interrupt flag is cleared*

Bits	R/W	Description
Bit 0	S-R reset pin	Unknown
Bit 1	S-R reset pin	Unknown
Bit 2	S-R flip-flop reset pin	Unknown Clear interrupt #12
Bit 3	S-R flip-flop reset pin	Unknown Clear interrupt #11
Bit 4	S-R flip-flop reset pin	Unknown Clear interrupt #10
Bit 5	S-R flip-flop reset pin	Unknown Clear interrupt #9
Bit 6	S-R reset pin	Unknown
Bit 7	S-R reset pin	Unknwon

## REGISTER 0x29 – Interrupt flag #15-#22

Start-up value: 0x00

### Description:

*Interrupt flag of events #15-#22.*

*When flag is set, the interrupt is constantly called until it's cleared.*

*If "0" is written to a bit, interrupt flag is ignored*

*If "1" is written to a bit, interrupt flag is cleared*

Bits	R/W	Description
Bit 0	S-R flip-flop reset pin	Press key "A" event Clear interrupt #22
Bit 1	S-R flip-flop reset pin	Press key "B" event Clear Interrupt #21
Bit 2	S-R flip-flop reset pin	Press key "C" event Clear Interrupt #20
Bit 3	S-R flip-flop reset pin	Press D-pad up key event Clear Interrupt #19
Bit 4	S-R flip-flop reset pin	Press D-pad down key event Clear Interrupt #18
Bit 5	S-R flip-flop reset pin	Press D-pad left key event Clear Interrupt #17
Bit 6	S-R flip-flop reset pin	Press D-pad right key event Clear Interrupt #16
Bit 7	S-R flip-flop reset pin	Press power button event Clear Interrupt #15

## REGISTER 0x2A – Interrupt flag #13-#14

Start-up value: 0x00

### Description:

*Interrupt flag of events #13-#14.*

*When flag is set, the interrupt is constantly called until it's cleared.*

*If "0" is written to a bit, interrupt flag is ignored*

*If "1" is written to a bit, interrupt flag is cleared*

Bits	R/W	Description
Bit 0 –5	S-R reset pin	Unknown
Bit 6	S-R flip-flop reset pin	Shock detector trigger Clear interrupt #14
Bit 7	S-R Flip-Flop reset pin	Unknown Clear interrupt #13

### **REGISTER 0x30 – Timer 1 control 1**

**Start-up value: 0x84**

**Description:**

*Part of timer 1 control.*

*NOTE: Lower 4-bits R/W access acts funny if bit 7 is '0'*

Bits	R/W	Description
Bit 0	Read/Write	Unknown
Bit 1	Unused	Unused
Bit 2	Write only	Preset timer
Bit 3 – 6	Unused	Unused
Bit 7	Read/Write	Enable timer 1 counting

### **REGISTER 0x31 – Timer 1 control 2**

**Start-up value: 0x00**

**Description:**

*Part of timer 1 control.*

*NOTE: Lower 4-bits R/W access acts funny if register 0x30 bit 7 is '0'*

Bits	R/W	Description
Bit 0 – 2	Unused	Unused
Bit 3	Read/Write	Unknown
Bit 4 – 7	Unused	Unused

### **REGISTER 0x32-0x33 – Timer 1 preset value**

**Start-up value: 0xF000**

**Description:**

*Timer 1 preset value, when timer 1 is reset or overflowed... counting values will automatically match this register.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	16-Bits preset value

### **REGISTER 0x34-0x35 – Timer 1 sound-pivot (unused)**

**Start-up value: 0x0000**

**Description:**

*These registers aren't used in hardware.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	Unused on hardware

## **REGISTER 0x36-0x37 – Timer 1 counter**

**Start-up value: 0xE33F**

**Description:**

*Return the counter value of timer 1.*

*NOTE: It's recommended to halt the counter while reading this register since an overflow of any byte may occur.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	16-Bits counter value

## **REGISTER 0x38 – Timer 2 control 1**

**Start-up value: 0x00**

**Description:**

*Part of timer 2 control.*

*NOTE: Lower 4-bits R/W access act funny if bit 7 is '0'*

Bits	R/W	Description
Bit 0	Read/Write	Unknown
Bit 1	Unused	Unused
Bit 2	Write Only	Preset Timer
Bit 3 – 6	Unused	Unused
Bit 7	Read/Write	Enable Timer 2 Counting

## **REGISTER 0x39 – Timer 2 control 2**

**Start-up value: 0x00**

**Description:**

*Part of timer 2 control.*

*NOTE: Lower 4-bits R/W access acts funny if register 0x38 bit 7 is '0'*

Bits	R/W	Description
Bit 0 – 2	Unused	Unused
Bit 3	Read/Write	Unknown
Bit 4 – 7	Unused	Unused



### **REGISTER 0x3A-0x3B – Timer 2 preset value**

**Start-up value:** 0xFFFF

**Description:**

*Timer 2 preset value, when timer 2 is reset or overflowed... counting values will automatically match this register.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	16-bits preset value

### **REGISTER 0x3C-0x3D – Timer 2 sound-pivot (unused)**

**Start-up value:** 0x0000

**Description:**

*These registers aren't used in hardware.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	Unused on hardware

### **REGISTER 0x3E-0x3F – Timer 2 counter**

**Start-up value:** 0xE3FF

**Description:**

*Return the counter value of timer 2.*

*NOTE: It's recommended to halt the counter while reading this register since an overflow of any Byte may occur.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	16-bits counter value

### **REGISTER 0x40 – 256 Hz timer control**

**Start-up value:** 0x00

**Description:**

*Controls 256 Hz timer.*

Bits	R/W	Description
Bit 0	Read/Write	0=Timer paused / 1=Timer running
Bit 1	Write Only	1=Timer reset (counter becomes 0x00)
Bit 2 – 7	Unused	Unused

### **REGISTER 0x41 – 256 Hz timer counter**

**Start-up value:** 0xCA

**Description:**

*Value of 8 low-bits of 256 Hz timer.*

Bits	R/W	Description
Bit 0 – 7	Read Only	8-bits 256 Hz timer counter

### **REGISTER 0x48 – Timer 3 control 1**

**Start-up value:** 0x84

**Description:**

*Part of timer 3 control.*

*NOTE: Lower 4-bits R/W access act funny if bit 7 is '0'*

Bits	R/W	Description
Bit 0	Read/Write	Unknown
Bit 1	Unused	Unused
Bit 2	Write Only	Preset timer
Bit 3 – 6	Unused	Unused
Bit 7	Read/Write	Enable timer 3 counting

## **REGISTER 0x49 – Timer 3 control 2**

**Start-up value:** 0x00

**Description:**

*Part of timer 3 control.*

*NOTE: Lower 4-bits R/W access act funny if register 0x48 bit 7 is '0'*

Bits	R/W	Description
Bit 0 – 2	Unused	Unused
Bit 3	Read/Write	Unknown
Bit 4 – 7	Unused	Unused

## **REGISTER 0x4A-0x4B – Timer 3 preset value**

**Start-up value:** 0x0BFE

**Description:**

*Timer 2 preset value, when timer 3 is reset or overflowed... Counting values will automatically match this register.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	16-bits preset value

## **REGISTER 0x4C-0x4D – Timer 3 sound-pivot**

**Start-up value:** 0x05FE

**Description:**

*This controls the pulse-width of playing sound.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	Sound-pivot location: Pulse-Width of 0% = 0x0000 Pulse-Width of 50% = Half of preset-value Pulse-Width of 100% = Same as preset-value

## **REGISTER 0x4E-0x4F – Timer 3 counter**

**Start-up value:** 0x8004

**Description:**

*Return the counter value of timer 3.*

*NOTE: It's recommended to halt the counter while reading this register since an overflow of any byte may occur.*

Bits	R/W	Description
Bit 0 – 15	Read/Write	16-Bits counter value

## **REGISTER 0x52 – Keypad status**

**Start-up value: 0xFF**

**Description:**

*Value changes when user press or release any key.*

*0 = Key pressed.*

*1 = Key released.*

Bits	R/W	Description
Bit 0	Read Only	Key "A"
Bit 1	Read Only	Key "B"
Bit 2	Read Only	Key "C"
Bit 3	Read Only	D-Pad up (directional-pad up)
Bit 4	Read Only	D-Pad down (directional-pad down)
Bit 5	Read Only	D-Pad left (directional-pad left)
Bit 6	Read Only	D-Pad right (directional-pad right)
Bit 7	Read Only	Power button

## **REGISTER 0x60 – I/O peripheral circuit select**

**Start-up value: 0x32**

**Description:**

*Activate a particular Peripheral.*

Bits	R/W	Description
Bit 0	Read/Write	Unknown, logic "0" recommended
Bit 1	Read/Write	IR receive/transmit
Bit 2	Read/Write	EEPROM / RTC data
Bit 3	Read/Write	EEPROM / RTC clock
Bit 4	Read/Write	Rumble controller
Bit 5	Read/Write	IR enable/disable
Bit 6	Read/Write	Unknown, logic "0" recommended
Bit 7	Read/Write	Unknown, logic "0" recommended

## **REGISTER 0x61 – I/O peripheral status control**

**Start-up value: 0x64**

**Description:**

*Control the peripherals, to select which one should be mapped, use register 0x60.*

Bits	R/W	Description
Bit 0	Read Only	IR received Bit If device not selected: read "0"
Bit 1	Read/Write	IR transmit Bit If device not selected: read "0"
Bit 2	Read/Write	EEPROM/RTC data If device not selected: read "1"
Bit 3	Read/Write	EEPROM/RTC clock If device not selected: read "0"
Bit 4	Read/Write	Rumble On/Off If device not selected: read "0"
Bit 5	Read/Write	IR disable (receive & transmit) If device not selected: read "1"
Bit 6	Unused	Always "1"
Bit 7	Read/Write	IR received bit (mirror) If device not selected: read "0"

## **REGISTER 0x71 – Sound volume**

**Start-up value: 0x00**

**Description:**

*Change the sound volume and do something else.*

Bits	R/W	Description
Bit 0 – 1	Read/Write	Sound Volume 0: 0% 1: 50% 2: 50% 3: 100%
Bit 2	Read/Write	DON'T SET TO "1" OR POKÉMON-MINI CRASHES
Bit 3 – 7	Unused	Unused

## **REGISTER 0x80 – LCD control**

**Start-up value: 0x00**

**Description:**

*Controls LCD.*

Bits	R/W	Description
Bit 0	Read/Write	Invert colors 0: Normal 1: Inverted
Bit 1	Read/Write	Enable rendering of background
Bit 2	Read/Write	Enable rendering of sprites
Bit 3	Read/Write	Global rendering enable
Bit 4 – 5	Read/Write	Map size: 0: 12x16 1: 16x12 2: 24x8 3: 24x8 (Prohibited code)
Bit 6 - 7	Unused	Unused

## **REGISTER 0x81 – LCD render refresh-rate**

**Start-up value: 0x00**

**Description:**

*Controls LCD.*

Bits	R/W	Description
Bit 0	Read/Write	Unknown
Bit 1 – 3	Read/Write	LCD refresh rate divider: 0: 60Hz / 3 = 20Hz (0 - 2) 1: 60Hz / 6 = 10Hz (0 - 5) 2: 60Hz / 9 = 6,6Hz (0 - 8) 3: 60Hz / 12 = 5 Hz (0 - B) 4: 60Hz / 2 = 30 Hz (0 - 1) 5: 60Hz / 4 = 15 Hz (0 - 3) 6: 60Hz / 6 = 10 Hz (0 - 5) 7: 60Hz / 8 = 7,5 Hz (0 - 7)
Bit 4 – 7	Read Only	Divider position, when overflow the LCD is updated

## **REGISTER 0x82-0x84 – BG tile data memory offset**

**Start-up value:** 0x000000

**Description:**

*Points to the location where the Background Rendering should load the tile-data from.*

*Lower 3 bits are always “0” (tiles aligned by 8 bytes)*

Bits	R/W	Description
Bit 0 – 2	Logic “0”	background tile-data memory offset aligned by 8 bytes
Bit 20 – 3	Read/Write	
Bit 23–21	Unused	Unused

## **REGISTER 0x85 – BG vertical move**

**Start-up value:** 0x00

**Description:**

*Move the background up.*

*NOTE: Writing outside the moving area doesn’t move the background but still affect the read-back value.*

Bits	R/W	Description
Bit 0 – 6	Read/Write	Move the background up, move range: Map size 0: 0x00 to 0x40 Map size 1: 0x00 to 0x20 Map size 2: move ignored
Bit 7	Unused	Unused

## **REGISTER 0x86 – BG horizontal move**

**Start-up value:** 0x00

**Description:**

*Move the background left.*

*NOTE: Writing outside the moving area doesn’t move the background but still affect the read-back value.*

Bits	R/W	Description
Bit 0 – 6	Read/Write	Move the background left, move range: Map Size 0: move ignored Map Size 1: 0x00 to 0x20 Map Size 2: 0x00 to 0x60
Bit 7	Unused	Unused

### **REGISTER 0x87-0x89 – Sprite tile data memory offset**

**Start-up value:** 0x000000

**Description:**

*Points to the location where the sprite rendering should load the tile-data.*

*Lower 6 bits are always “0” (tiles aligned by 64 bytes)*

Bits	R/W	Description
Bit 0 – 5	Logic “0”	Sprite tile data memory offset aligned by 64 bytes
Bit 20 – 6	Read/Write	
Bit 23–21	Unused	Unused

### **REGISTER 0x8A – LCD status**

**Start-up value:** 0x00

**Description:**

*Receive LCD status.*

Bits	R/W	Description
Bit 0	Read Only	Unknown
Bit 1	Read Only	Unknown
Bit 2	Read Only	Unknown
Bit 3	Read Only	Unknown
Bit 4	Read Only	LCD during V-Sync
Bit 5	Read Only	Unknown
Bit 6 – 7	Unused	Unused

### **REGISTER 0xFE – Direct-LCD control / data**

**Start-up value:** 0x40

**Description:**

*Controls directly the LCD.*

Bits	R/W	Description
Bit 0 – 7	Read/Write	Direct-LCD command or data



## **REGISTER 0xFF – Direct-LCD data**

**Start-up value: 0xFF**

**Description:**

*Controls directly the LCD.*

Bits	R/W	Description
Bit 0 – 7	Read/Write	Direct-LCD data

## **Known Direct-LCD commands:**

**Command value: 0x81**

**Description:**

*Change LCD contrast.*

REG 0xFE	REG 0xFE or 0xFF
0x81	Contrast Intensity

*Contrast Intensity:*

*0x00 (white on white)*

*0x1F (normal)*

*0x3F (black on black)*

## 6. Video processing

PM (Pokémon-Mini) LCD with a resolution of 96x64x1bpp is capable to render 1 Background (8x8 Pixels Blocks Tiles) and 24 16x16 Sprites.

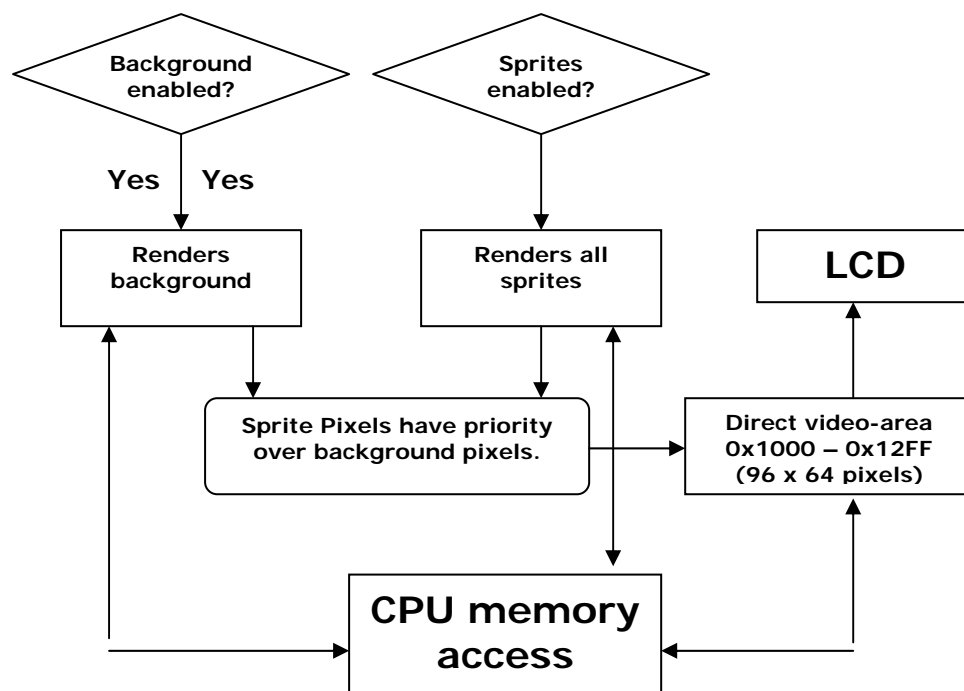
Background is able to move (not scroll) within the viewing area and invert pixels (white turns black and black turns white).

Each sprite can freely move, has his own Mask, can Flip Horizontally & Vertically, has a enable flag and supports inverting pixels.

The rendering system supports a variable refresh rate (up to 30 Hz maximum).



By default, Pokémon-Mini displays the content of 0x1000-0x12FF directly to the LCD but if any or both sprites & background Hardware Rendering are enabled, 0x1000-0x12FF get overwritten by the renderer.



## Direct video area (0x1000-0x12FF)

Organized into sectors of 96x8 Pixels (8 Sectors), each Byte represents a Column of 8 Pixels (from top to bottom) with 96 Pixels Width.

**96 x 64 = 6144 Pixels = 768 Bytes**

/	0	1	2	3	4	5	6	7	8	9	10	11	...	95
0	0	8	16	24	32	40	48	56	64	72	80	88	...	760
1	1	9	17	25	33	41	49	57	65	73	81	89	...	761
2	2	10	18	26	34	42	50	58	66	74	82	90	...	762
3	3	11	19	27	35	43	51	59	67	75	83	91	...	763
4	4	12	20	28	36	44	52	60	68	76	84	92	...	764
5	5	13	21	29	37	45	53	61	69	77	85	93	...	765
6	6	14	22	30	38	46	54	62	70	78	86	94	...	766
7	7	15	23	31	39	47	55	63	71	79	87	95	...	767
8	768	776	784	792	800	808	816	824	832	840	848	856	...	1528
9	769	777	785	793	801	809	817	825	833	841	849	857	...	1529
10	770	778	786	794	802	810	818	826	834	842	850	858	...	1530
11	771	779	787	795	803	811	819	827	835	843	851	859	...	1531
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
63	5383	5391	5399	5407	5415	5423	5431	5439	5447	5455	5463	5471	...	6143

Bit location = (Returned number from table)  
 Byte location = (Returned number from table) / 8

It's unknown if CPU Stops when trying to write this area while being outside V-Blank period.

## Background Display

BG renderer can display 12x16, 16x12 or 24x8 tiles of 8x8 pixels into the screen; the field can be moved via REG. 0x85 (vertical) and 0x86 (horizontal) although it doesn't warp around.

Tile-data is pointed at REG. 0x82 to 0x84 and can be anywhere within the full-range Pokémon-Mini memory area (0x00000 to 0xFFFFF)

Tile-map is at 0x1360 and each byte stands the tile-ID from left-top to bottom-right (the way it's mapped depends of the map size selected).

Pokémon-Mini renderer supports up to 256 different tiles at same time (But only 96 are visible due to maximum in-screen size).

Each tile is aligned 8 Bytes (8x8 Pixels with 1bpp depth).

**EXAMPLE of map size 16x12 without any vertical or horizontal moving (Only first 12x8 are visible):**

/	0	1	2	3	4	5	6	7	8	9	10	11	...	15
0	0	1	2	3	4	5	6	7	8	9	10	11	...	15
1	16	17	18	19	20	21	22	23	24	25	26	27	...	31
2	32	33	34	35	36	37	38	39	40	41	42	43	...	47
3	48	49	50	51	52	53	54	55	56	57	58	59	...	63
4	64	65	66	67	68	69	70	71	72	73	74	75	...	79
5	80	81	82	83	84	85	86	87	88	89	90	91	...	95
6	96	97	98	99	100	101	102	103	104	105	106	107	...	111
7	112	113	114	115	116	117	118	119	120	121	122	123	...	127
8	128	129	130	131	132	133	134	135	136	137	138	139	...	143

Tile-ID = MEMORY [0x1360 + (Returned number from table)]

1<sup>st</sup> Byte of Tile = MEMORY [REG 0x82-0x85 + (Tile-ID \* 8)]

**Each Tile (8x8 pixels):**

/	0	1	2	3	4	5	6	7
0	0	8	16	24	32	40	48	56
1	1	9	17	25	33	41	49	57
2	2	10	18	26	34	42	50	58
3	3	11	19	27	35	43	51	59
4	4	12	20	28	36	44	52	60
5	5	13	21	29	37	45	53	61
6	6	14	22	30	38	46	54	62
7	7	15	23	31	39	47	55	63

Bit Offset = (Returned number from table)

Byte Offset = (Returned number from table) / 8

**BIT (pixel):**

"0" = white (black if inverted)

"1" = black (white if inverted)

## Sprites Display

Sprites renderer can display up to twenty-four 16x16 sprites at same time into the screen.

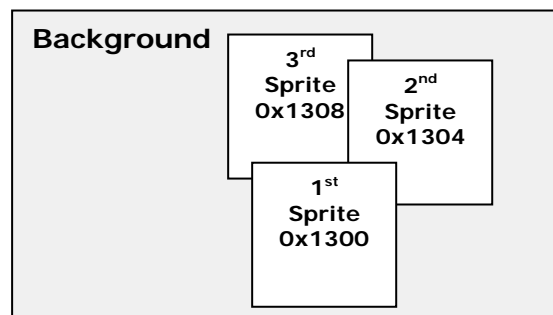
Sprite data is pointed at REG. 0x87 to 0x89 and can be anywhere into the full-range Pokémon-Mini memory area (0x00000 to 0xFFFFF)

Pokémon-Mini renderer supports up to 256 different sprites at same time (but only 24 are visible due to hardware limitation).

Each tile is aligned 64 bytes (16x16 pixels + 16x16 mask with 1bpp depth).

Sprite OAM area is located at 0x1300 and has 96 Bytes (24 sprites)

Note that all sprites have priority over background and the 1<sup>st</sup> sprite in the list is over all others.



0x1300...0x1303 – 1<sup>st</sup> Sprite

0x1304...0x1307 – 2<sup>nd</sup> Sprite

0x1308...0x130B – 3<sup>rd</sup> Sprite

0x130C...0x130F – 4<sup>th</sup> Sprite

(Etc...)

Byte 0:

< X Position on the screen minus 16 >

When value is 24, it displays 8 pixels to the left

Bit 7 is ignored

Byte 1:

< Y Position on the screen minus 16 >

When value is 24, it displays 8 pixels to the top

Bit 7 is ignored

Byte 2:

< Sprite-ID (0 to 255) >

1<sup>st</sup> Byte of Sprite = MEMORY [REG 0x87-0x89 + (Sprite-ID \* 64)]

Byte 3:

< Flags/configuration >

Bit 0: Flip-horizontally

Bit 1: Flip-vertically

Bit 2: Invert pixels colors (Doesn't affect MASK)

Bit 3: Enable sprite display

Bit 4: Ignored

Bit 5: Ignored

Bit 6: Ignored

Bit 7: Ignored

Sprites aren't mapped 16x16 but two 8x16 glued together horizontally.  
 There's an order of Tiles inside a sprite if there's no horizontal or vertical flipping (8x8 tiles):

[1 <sup>st</sup> ]	[5 <sup>th</sup> ]
[2 <sup>nd</sup> ]	[6 <sup>th</sup> ]
[3 <sup>rd</sup> ]	[7 <sup>th</sup> ]
[4 <sup>th</sup> ]	[8 <sup>th</sup> ]

White square = Draw

Dark square = Mask

$VRAM = (VRAM \& MASK) | (DRAW \& \sim MASK);$

#### **DRAW:**

"0" = white (black if inverted)

"1" = black (white if inverted)

#### **MASK:**

"0" = opaque

"1" = transparent

**To be completed....**