

---

Everything You Always Wanted To Know About GAMEBOY \*

---

\* but were afraid to ask

Pan of -ATX- Document Updated by contributions from:  
Marat Fayzullin, Pascal Felber, Paul Robson, Martin Korth

Last update 17-Mar-98 by kOOPa

Forward: The following was typed up for informational purposes regarding the inner workings on the hand-held game machine known as GameBoy, manufactured and designed by Nintendo Co., LTD. This info is presented to inform a user on how their Game Boy works and what makes it "tick". GameBoy is copyrighted by Nintendo Co., LTD. Any reference to copyrighted material is not presented for monetary gain, but for educational purposes and higher learning.

Terms

-----

GB = Original GameBoy

GBP = GameBoy Pocket/GameBoy Light

GBC = GameBoy Colo(u)r

SGB = Super GameBoy

## Game Boy Specs

-----

CPU: 8-bit (Similar to the Z80 processor.)

Main RAM: 8K Byte

Video RAM: 8K Byte

Screen Size 2.6"

Resolution: 160x144 (20x18 tiles)

Max # of sprites: 40

Max # sprites/line: 10

Max sprite size: 8x16

Min sprite size: 8x8

Clock Speed: 4.194304 MHz (4.295454 SGB, 4.194/8.388MHz GBC)

Horiz Sync: 9198 KHz (9420 KHz for SGB)

Vert Sync: 59.73 Hz (61.17 Hz for SGB)

Sound: 4 channels with stereo sound

Power: DC6V 0.7W (DC3V 0.7W for GB Pocket)

## Processor

-----

The GameBoy uses a computer chip similar to an Intel 8080. It contains all of the instructions of an 8080 except there are no exchange instructions. In many ways the processor is more similar to the Zilog Z80 processor. Compared to the Z80, some instructions have been added and some have been

taken away.

The following are added instructions:

```
ADD  SP,nn                ;nn = signed byte
LD   (HLI),A              ;Write A to (HL) and increment HL
LD   (HLD),A              ;Write A to (HL) and decrement HL
LD   A,(HLI)              ;Write (HL) to A and increment HL
LD   A,(HLD)              ;Write (HL) to A and decrement HL
LD   A,($FF00+nn)
LD   A,($FF00+C)
LD   ($FF00+nn),A
LD   ($FF00+C),A
LD   (nnnn),SP
LD   HL,SP+nn             ;nn = signed byte
STOP                                ;Stop processor & screen until button press
SWAP r                     ;Swap high & low nibbles of r
```

The following instructions have been removed:

Any command that uses the IX or IY registers.

All IN/OUT instructions.

All exchange instructions.

All commands prefixed by ED (except remapped RETI).

All conditional jumps/calls/rets on parity/overflow and sign flag.

The following instructions have different opcodes:

```
LD   A,[nnnn]
```

LD [nnnn],A

RETI

#### GB General Memory Map\*

-----

Interrupt Enable Register

----- FFFF

Internal RAM

----- FF80

Empty but unusable for I/O

----- FF4C

I/O ports

----- FF00

Empty but unusable for I/O

----- FEA0

Sprite Attrib Memory (OAM)

----- FE00

Echo of 8kB Internal RAM

----- E000

8kB Internal RAM

----- C000

8kB switchable RAM bank

----- A000

8kB Video RAM

----- 8000

32kB Cartridge

----- 0000

\* NOTE: b = bit, B = byte

Echo of 8kB Internal RAM

-----

The addresses E000-FE00 appear to access the internal RAM the same as C000-DE00. (i.e. If you write a byte to address E000 it will appear at C000 and E000. Similarly, writing a byte to C000 will appear at C000 and E000.)

User I/O

-----

There are no empty spaces in the memory map for implementing input ports except the switchable RAM bank area (not an option on the Super Smart Card since it's RAM bank is always enabled).

An output only port may be implemented anywhere between A000-FDFF. If implemented in a RAM area care should be taken to use an area of RAM not used for anything else. (FE00 and above can't be used because the CPU doesn't generate an external /WR for these locations.)

If you have a cart with an MBC1, a ROM 4Mbit or smaller, and a RAM 8Kbyte or smaller (or no RAM) then you can use pins 6 & 7 of the MBC1 for 2 digital output pins for whatever purpose you wish. To use them you must first put the MBC1 into 4MbitROM/32KbyteRAM mode by writing 01 to 6000. The two least significant bits you write to 4000 will then be output to these pins.

#### Reserved Memory Locations

-----

0000	Restart \$00 Address (RST \$00 calls this address.)
0008	Restart \$08 Address (RST \$08 calls this address.)
0010	Restart \$10 Address (RST \$10 calls this address.)
0018	Restart \$18 Address (RST \$18 calls this address.)
0020	Restart \$20 Address (RST \$20 calls this address.)
0028	Restart \$28 Address (RST \$28 calls this address.)
0030	Restart \$30 Address (RST \$30 calls this address.)
0038	Restart \$38 Address (RST \$38 calls this address.)

0040        Vertical Blank Interrupt Start Address

0048        LCDC Status Interrupt Start Address

0050        Timer Overflow Interrupt Start Address

0058        Serial Transfer Completion Interrupt Start Address

0060        High-to-Low of P10-P13 Interrupt Start Address

An internal information area is located at 0100-014F in each cartridge. It contains the following values:

0100-0103    This is the begin code execution point in a cart. Usually there is a NOP and a JP instruction here but not always.

0104-0133    Scrolling Nintendo graphic:

CE ED 66 66 CC 0D 00 0B 03 73 00 83 00 0C 00 0D

00 08 11 1F 88 89 00 0E DC CC 6E E6 DD DD D9 99

BB BB 67 63 6E 0E EC CC DD DC 99 9F BB B9 33 3E

( PROGRAM WON'T RUN IF CHANGED!!!)

0134-0142    Title of the game in UPPER CASE ASCII. If it is less than 16 characters then the remaining bytes are filled with 00's.

0143        \$80 = Color GB, \$00 or other = not Color GB

0144      Ascii hex digit, high nibble of licensee code (new).

0145      Ascii hex digit, low nibble of licensee code (new).

          (These are normally \$00 if [\$014B] <> \$33.)

0146      GB/SGB Indicator (00 = GameBoy, 03 = Super GameBoy functions)

          (Super GameBoy functions won't work if <> \$03.)

0147      Cartridge type:

0 - ROM ONLY	12 - ROM+MBC3+RAM
1 - ROM+MBC1	13 - ROM+MBC3+RAM+BATT
2 - ROM+MBC1+RAM	19 - ROM+MBC5
3 - ROM+MBC1+RAM+BATT	1A - ROM+MBC5+RAM
5 - ROM+MBC2	1B - ROM+MBC5+RAM+BATT
6 - ROM+MBC2+BATTERY	1C - ROM+MBC5+RUMBLE
8 - ROM+RAM	1D - ROM+MBC5+RUMBLE+SRAM
9 - ROM+RAM+BATTERY	1E - ROM+MBC5+RUMBLE+SRAM+BATT
B - ROM+MMM01	1F - Pocket Camera
C - ROM+MMM01+SRAM	FD - Bandai TAMA5
D - ROM+MMM01+SRAM+BATT	FE - Hudson HuC-3
F - ROM+MBC3+TIMER+BATT	FF - Hudson HuC-1
10 - ROM+MBC3+TIMER+RAM+BATT	
11 - ROM+MBC3	

0148      ROM size:

0 - 256Kbit = 32KByte = 2 banks
1 - 512Kbit = 64KByte = 4 banks
2 - 1Mbit = 128KByte = 8 banks
3 - 2Mbit = 256KByte = 16 banks



4 - 4Mbit = 512KByte = 32 banks  
5 - 8Mbit = 1MByte = 64 banks  
6 - 16Mbit = 2MByte = 128 banks  
\$52 - 9Mbit = 1.1MByte = 72 banks  
\$53 - 10Mbit = 1.2MByte = 80 banks  
\$54 - 12Mbit = 1.5MByte = 96 banks

0149 RAM size:

0 - None  
1 - 16kBit = 2kB = 1 bank  
2 - 64kBit = 8kB = 1 bank  
3 - 256kBit = 32kB = 4 banks  
4 - 1MBit = 128kB = 16 banks

014A Destination code:

0 - Japanese  
1 - Non-Japanese

014B Licensee code (old):

33 - Check 0144/0145 for Licensee code.  
79 - Accolade  
A4 - Konami  
(Super GameBoy function won't work if <> \$33.)

014C Mask ROM Version number (Usually \$00)

014D Complement check

(PROGRAM WON'T RUN ON GB IF NOT CORRECT!!!)  
(It will run on Super GB, however, if incorrect.)

014E-014F   Checksum (higher byte first) produced by  
              adding all bytes of a cartridge except for two  
              checksum bytes and taking two lower bytes of  
              the result. (GameBoy ignores this value.)

#### Cartridge Types

-----

The following define the byte at cart location 0147:

#### ROM ONLY

This is a 32kB (256kb) ROM and occupies 0000-7FFF.

#### MBC1 (Memory Bank Controller 1)

MBC1 has two different maximum memory modes:

16Mbit ROM/8KByte RAM   or   4Mbit ROM/32KByte RAM.

The MBC1 defaults to 16Mbit ROM/8KByte RAM mode on power up. Writing a value (XXXXXXXS - X = Don't care, S = Memory model select) into 6000-7FFF area will select the memory model to use. S = 0 selects 16/8 mode. S = 1 selects 4/32 mode.

Writing a value (XXXBBBBB - X = Don't cares, B = bank select bits) into 2000-3FFF area will select an appropriate ROM bank at 4000-7FFF. Values of 0 and 1

do the same thing and point to ROM bank 1. Rom bank 0 is not accessible from 4000-7FFF and can only be read from 0000-3FFF.

If memory model is set to 4/32:

Writing a value (XXXXXXBB - X = Don't care, B = bank select bits) into 4000-5FFF area will select an appropriate RAM bank at A000-C000. Before you can read or write to a RAM bank you have to enable it by writing a XXXX1010 into 0000-1FFF area\*. To disable RAM bank operations write any value but XXXX1010 into 0000-1FFF area. Disabling a RAM bank probably protects that bank from false writes during power down of the GameBoy. (NOTE: Nintendo suggests values \$0A to enable and \$00 to disable RAM bank!!)

If memory model is set to 16/8 mode:

Writing a value (XXXXXXBB - X = Don't care, B = bank select bits) into 4000-5FFF area will set the two most significant ROM address lines.

\* NOTE: The Super Smart Card doesn't require this operation because it's RAM bank is ALWAYS enabled. Include this operation anyway to allow your code to work with both.

MBC2 (Memory Bank Controller 2):

This memory controller works much like the MBC1 controller with the following exceptions:

MBC2 will work with ROM sizes up to 2Mbit.

Writing a value (XXXXBBBB - X = Don't cares, B = bank select bits) into 2100-21FF area will select an appropriate ROM bank at 4000-7FFF.

RAM switching is not provided. Unlike the MBC1 which uses external RAM, MBC2 has 512 x 4 bits of RAM which is in the controller itself. It still requires an external battery to save data during power-off though.

The least significant bit of the upper address byte must be zero to enable/disable cart RAM. For example the following addresses can be used to enable/disable cart RAM:

0000-00FF, 0200-02FF, 0400-04FF, ..., 1E00-1EFF.

The suggested address range to use for MBC2 ram enable/disable is 0000-00FF.

The least significant bit of the upper address byte must be one to select a ROM bank. For example the following addresses can be used to select a ROM bank:

2100-21FF, 2300-23FF, 2500-25FF, ..., 3F00-3FFF.

The suggested address range to use for MBC2 rom bank selection is 2100-21FF.

MBC3 (Memory Bank Controller 3):

This controller is similar to MBC1 except it accesses all 16mbits of ROM without requiring any writes to the 4000-5FFF area.

Writing a value (XBBBBBBB - X = Don't care, B = bank select bits) into 2000-3FFF area will select an appropriate ROM bank at 4000-7FFF.

Also, this MBC has a built-in battery-backed Real Time Clock (RTC) not found in any other MBC. Some MBC3 carts do not support it (WarioLand II non-color version) but some do (Harvest Moon/Japanese version.)

#### MBC5 (Memory Bank Controller 5):

This controller is the first MBC that is guaranteed to run in GameBoy Color double-speed mode but it appears the other MBC's run fine in GBC double-speed mode as well.

It is similar to the MBC3 (but no RTC) but can access up to 64mbits of ROM and up to 1mbit of RAM. The lower 8 bits of the 9-bit rom bank select is written to the 2000-2FFF area while the upper bit is written to the least significant bit of the 3000-3FFF area.

Writing a value (XXXXBBBB - X = Don't care, B = bank select bits) into 4000-5FFF area will select an appropriate RAM bank at A000-BFFF if the cart contains RAM. Ram sizes are 64kbit, 256kbit, & 1mbit.

Also, this is the first MBC that allows rom bank 0 to appear in the 4000-7FFF range by writing \$000 to the rom bank select.

#### Rumble Carts:

Rumble carts use an MBC5 memory bank controller. Rumble carts can only have up to 256kbits of RAM. The highest RAM address line that allows 1mbit of RAM on MBC5 non-rumble carts is used as the motor on/off for the rumble cart.

Writing a value (XXXXM BBB - X = Don't care, M = motor, B = bank select bits) into 4000-5FFF area will select an appropriate RAM bank at A000-BFFF if the cart contains RAM. RAM sizes are 64kbit or 256kbits. To turn the rumble motor on set M = 1, M = 0 turns it off.

#### HuC1 (Memory Bank / Infrared Controller):

This controller made by Hudson Soft appears to be very similar to an MBC1 with the main difference being that it supports infrared LED input / output. The Japanese cart "Fighting Phoenix" (internal cart name: SUPER B DAMAN) is known to contain this chip.

#### Bung Carts:

The flash carts sold by Bung (<http://www.bung.com.hk>)

on power up appear like an MBC5 cart except that writing \$00 to the bank select register selects rom bank 1 instead of rom bank 0. Writing values to 3000-3FFF area does nothing.

Writing values (XXXXXXS - X = Don't care, S = Select) to 6000-7FFF area selects MBC1 16/8 mode if S=0 except that writes to 3000-3FFF area still do nothing. If S=1 then MBC5 mode is selected. (S=1 by default on power up.)

Cart locations A000 & A100 act as special write only hardware control registers if a value of \$c0 is written to 0000-1FFF area. These hardware control registers are used for setting up cart hardware for different games when the Bung cart is used as a Multicart for holding several different games.

#### Power Up Sequence

-----

When the GameBoy is powered up, a 256 byte program starting at memory location 0 is executed. This program is located in a ROM inside the GameBoy. The first thing the program does is read the cartridge locations from \$104 to \$133 and place this graphic of a Nintendo logo on the screen at the top. This image is then scrolled

until it is in the middle of the screen. Two musical notes are then played on the internal speaker. Again, the cartridge locations \$104 to \$133 are read but this time they are compared with a table in the internal rom. If any byte fails to compare, then the GameBoy stops comparing bytes and simply halts all operations.

GB & GB Pocket:

Next, the GameBoy starts adding all of the bytes in the cartridge from \$134 to \$14d. A value of 25 decimal is added to this total. If the least significant byte of the result is a not a zero, then the GameBoy will stop doing anything.

Super GB:

Even though the GB & GBP check the memory locations from \$134 to \$14d, the SGB doesn't.

If the above checks pass then the internal ROM is disabled and cartridge program execution begins at location \$100 with the following register values:

AF=\$01-GB/SGB, \$FF-GBP, \$11-GBC

F =\$B0

B =\$00-GB/SGB/GBP/GBC, \$01-GBA

C =\$13

DE=\$00D8

HL=\$014D



Stack Pointer=\$FFFE

[\$FF05] = \$00 ; TIMA

[\$FF06] = \$00 ; TMA

[\$FF07] = \$00 ; TAC

[\$FF10] = \$80 ; NR10

[\$FF11] = \$BF ; NR11

[\$FF12] = \$F3 ; NR12

[\$FF14] = \$BF ; NR14

[\$FF16] = \$3F ; NR21

[\$FF17] = \$00 ; NR22

[\$FF19] = \$BF ; NR24

[\$FF1A] = \$7F ; NR30

[\$FF1B] = \$FF ; NR31

[\$FF1C] = \$9F ; NR32

[\$FF1E] = \$BF ; NR33

[\$FF20] = \$FF ; NR41

[\$FF21] = \$00 ; NR42

[\$FF22] = \$00 ; NR43

[\$FF23] = \$BF ; NR30

[\$FF24] = \$77 ; NR50

[\$FF25] = \$F3 ; NR51

[\$FF26] = \$F1-GB, \$F0-SGB ; NR52

[\$FF40] = \$91 ; LCDC

[\$FF42] = \$00 ; SCY

[\$FF43] = \$00 ; SCX

[\$FF45] = \$00 ; LYC

[\$FF47] = \$FC ; BGP

[\$FF48] = \$FF ; OBP0

[\$FF49] = \$FF ; OBP1

```
[$FF4A] = $00    ; WY
[$FF4B] = $00    ; WX
[$FFFF] = $00    ; IE
```

It is not a good idea to assume the above values will always exist. A later version GameBoy could contain different values than these at reset. Always set these registers on reset rather than assume they are as above.

Please note that GameBoy internal RAM on power up contains random data. All of the GameBoy emulators tend to set all RAM to value \$00 on entry.

Cart RAM the first time it is accessed on a real GameBoy contains random data. It will only contain known data if the GameBoy code initializes it to some value.

Stop Mode

-----

The STOP command halts the GameBoy processor and screen until any button is pressed. The GB and GBP screen goes white with a single dark horizontal line. The GBC screen goes black.

## Low-Power Mode

-----

It is recommended that the HALT instruction be used whenever possible to reduce power consumption & extend the life of the batteries. This command stops the system clock reducing the power consumption of both the CPU and ROM.

The CPU will remain suspended until an interrupt occurs at which point the interrupt is serviced and then the instruction immediately following the HALT is executed. If interrupts are disabled (DI) then halt doesn't suspend operation but it does cause the program counter to stop counting for one instruction on the GB,GBP, and SGB as mentioned below.

Depending on how much CPU time is required by a game, the HALT instruction can extend battery life anywhere from 5 to 50% or possibly more.

WARNING: The instruction immediately following the HALT instruction is "skipped" when interrupts are disabled (DI) on the GB,GBP, and SGB. As a result, always put a NOP after the HALT instruction. This instruction skipping doesn't occur when interrupts are enabled (EI).

This "skipping" does not seem to occur on the GameBoy Color even in regular GB mode. (\$143=\$00)

EXAMPLES from Martin Korth who documented this problem:

(assuming interrupts disabled for all examples)

- 1) This code causes the 'a' register to be incremented TWICE.

```
76          halt
3C          inc  a
```

- 2) The next example is a bit more difficult. The following code

```
76          halt
FA 34 12    ld   a, (1234)
```

is effectively executed as

```
76          halt
FA FA 34    ld   a, (34FA)
12          ld   (de), a
```

- 3) Finally an interesting side effect

```
76          halt
76          halt
```

This combination hangs the cpu.

The first HALT causes the second HALT to be repeated, which therefore causes the following command (=itself) to be repeated - again and again.

Placing a NOP between the two halts would cause the NOP to be repeated once, the second HALT wouldn't lock the cpu.

Below is suggested code for GameBoy programs:

```
; **** Main Game Loop ****
```

```
Main:
```

```
    halt                ; stop system clock
                        ; return from halt when interrupted

    nop                ; (See WARNING above.)

    ld     a, (VblnkFlag)
    or     a            ; V-Blank interrupt ?
    jr     z, Main      ; No, some other interrupt

    xor     a
    ld     (VblnkFlag), a ; Clear V-Blank flag

    call    Controls    ; button inputs
    call    Game        ; game operation

    jr     Main
```

```
; **** V-Blank Interrupt Routine ****
```

```
Vblnk:
```

```
    push    af
    push    bc
    push    de
    push    hl

    call    SpriteDma    ; Do sprite updates
```

```

ld      a,1

ld      (VblnkFlag),a


pop     hl
pop     de
pop     bc
pop     af
reti

```

## Video

-----

The main GameBoy screen buffer (background) consists of 256x256 pixels or 32x32 tiles (8x8 pixels each). Only 160x144 pixels can be displayed on the screen. Registers SCROLLX and SCROLLY hold the coordinates of background to be displayed in the left upper corner of the screen. Background wraps around the screen (i.e. when part of it goes off the screen, it appears on the opposite side.)

An area of VRAM known as Background Tile Map contains the numbers of tiles to be displayed. It is organized as 32 rows of 32 bytes each. Each byte contains a number of a tile to be displayed. Tile patterns are taken from the Tile Data Table located either at \$8000-8FFF or \$8800-97FF. In the first case, patterns are numbered with

unsigned numbers from 0 to 255 (i.e. pattern #0 lies at address \$8000). In the second case, patterns have signed numbers from -128 to 127 (i.e. pattern #0 lies at address \$9000). The Tile Data Table address for the background can be selected by setting the LCDC register.

There are two different Background Tile Maps. One is located from \$9800-9Bff. The other from \$9C00-9FFF. Only one of these can be viewed at any one time. The currently displayed background can be selected by setting the LCDC register.

Besides background, there is also a "window" overlaying the background. The window is not scrollable i.e. it is always displayed starting from its left upper corner. The location of a window on the screen can be adjusted via WNDPOSX and WNDPOSY registers. Screen coordinates of the top left corner of a window are WNDPOSX-7, WNDPOSY. The tile numbers for the window are stored in the Tile Data Table. None of the windows tiles are ever transparent. Both the Background and the window share the same Tile Data Table.

Both background and window can be disabled or enabled separately via bits in the LCDC register.

If the window is used and a scan line interrupt disables it (either by writing to LCDC or by setting WX > 166) and a scan line interrupt a little later on enables it

then the window will resume appearing on the screen at the exact position of the window where it left off earlier. This way, even if there are only 16 lines of useful graphics in the window, you could display the first 8 lines at the top of the screen and the next 8 lines at the bottom if you wanted to do so.

WX may be changed during a scan line interrupt (to either cause a graphic distortion effect or to disable the window (WX>166) ) but changes to WY are not dynamic and won't be noticed until the next screen redraw.

The tile images are stored in the Tile Pattern Tables. Each 8x8 image occupies 16 bytes, where each 2 bytes represent a line:

Tile:	Image:
.33333..	.33333.. -> 01111100 -> \$7C
22...22.	01111100 -> \$7C
11...11.	22...22. -> 00000000 -> \$00
2222222. <-- digits	11000110 -> \$C6
33...33. represent	11...11. -> 11000110 -> \$C6
22...22. color	00000000 -> \$00
11...11. numbers	2222222. -> 00000000 -> \$00
.....	11111110 -> \$FE
	33...33. -> 11000110 -> \$C6
	11000110 -> \$C6



```

22...22. -> 00000000 -> $00
           11000110 -> $C6
11...11. -> 11000110 -> $C6
           00000000 -> $00
..... -> 00000000 -> $00
           00000000 -> $00

```

As it was said before, there are two Tile Pattern Tables at \$8000-8FFF and at \$8800-97FF. The first one can be used for sprites, the background, and the window display. Its tiles are numbered from 0 to 255. The second table can be used for the background and the window display and its tiles are numbered from -128 to 127.

## Sprites

-----

GameBoy video controller can display up to 40 sprites either in 8x8 or in 8x16 pixels. Because of a limitation of hardware, only ten sprites can be displayed per scan line. Sprite patterns have the same format as tiles, but they are taken from the Sprite Pattern Table located at \$8000-8FFF and have unsigned numbering. Sprite attributes reside in the Sprite Attribute Table (OAM - Object Attribute Memory) at \$FE00-FE9F. OAM is divided into 40 4-byte blocks each of which corresponds to a sprite.

In 8x16 sprite mode, the least significant bit of the

sprite pattern number is ignored and treated as 0.

When sprites with different x coordinate values overlap, the one with the smaller x coordinate (closer to the left) will have priority and appear above any others.

When sprites with the same x coordinate values overlap, they have priority according to table ordering. (i.e. \$FE00 - highest, \$FE04 - next highest, etc.)

Please note that Sprite X=0, Y=0 hides a sprite. To display a sprite use the following formulas:

$\text{SpriteScreenPositionX}(\text{Upper left corner of sprite}) = \text{SpriteX} - 8$

$\text{SpriteScreenPositionY}(\text{Upper left corner of sprite}) = \text{SpriteY} - 16$

To display a sprite in the upper left corner of the screen set sprite X=8, Y=16.

Only 10 sprites can be displayed on any one line. When this limit is exceeded, the lower priority sprites (priorities listed above) won't be displayed. To keep unused sprites from affecting onscreen sprites set their Y coordinate to Y=0 or Y=>144+16. Just setting the X coordinate to X=0 or X=>160+8 on a sprite will hide it but it will still affect other sprites sharing the same lines.

Blocks have the following

format:

Byte0 Y position on the screen

Byte1 X position on the screen

Byte2 Pattern number 0-255 (Unlike some tile  
numbers, sprite pattern numbers are unsigned.  
LSB is ignored (treated as 0) in 8x16 mode.)

Byte3 Flags:

Bit7 Priority

If this bit is set to 0, sprite is displayed  
on top of background & window. If this bit  
is set to 1, then sprite will be hidden behind  
colors 1, 2, and 3 of the background & window.  
(Sprite only prevails over color 0 of BG & win.)

Bit6 Y flip

Sprite pattern is flipped vertically if  
this bit is set to 1.

Bit5 X flip

Sprite pattern is flipped horizontally if  
this bit is set to 1.

Bit4 Palette number

Sprite colors are taken from OBJ1PAL if  
this bit is set to 1 and from OBJ0PAL  
otherwise.

Sprite RAM Bug

-----

There is a flaw in the GameBoy hardware that causes trash to be written to OAM RAM if the following commands are used while their 16-bit content is in the range of \$FE00 to \$FEFF:

```
inc xx      (xx = bc,de, or hl)
dec xx
```

```
ldi a,(hl)
ldd a,(hl)
```

```
ldi (hl),a
ldd (hl),a
```

Only sprites 1 & 2 (\$FE00 & \$FE04) are not affected by these instructions.

Sound

-----

There are two sound channels connected to the output terminals SO1 and SO2. There is also a input terminal Vin connected to the cartridge. It can be routed to either of both output terminals. GameBoy circuitry allows producing sound in four different ways:

Quadrangular wave patterns with sweep and envelope functions.

Quadrangular wave patterns with envelope functions.

Voluntary wave patterns from wave RAM.

White noise with an envelope function.

These four sounds can be controlled independantly and then mixed separately for each of the output terminals.

Sound registers may be set at all times while producing sound.

When setting the initial value of the envelope and restarting the length counter, set the initial flag to 1 and initialize the data.

Under the following situations the Sound ON flag is reset and the sound output stops:

1. When the sound output is stopped by the length counter.
2. When overflow occurs at the addition mode while sweep is operating at sound 1.

When the Sound OFF flag for sound 3 (bit 7 of NR30) is set at 0, the cancellation of the OFF mode must be done by setting the sound OFF flag to 1. By initializing sound 3, it starts it's function.

When the All Sound OFF flag (bit 7 of NR52) is set to 0,

the mode registers for sounds 1,2,3, and 4 are reset and the sound output stops. (NOTE: The setting of each sounds mode register must be done after the All Sound OFF mode is cancelled. During the All Sound OFF mode, each sound mode register cannot be set.)

NOTE: DURING THE ALL SOUND OFF MODE, GB POWER CONSUMPTION DROPS BY 16% OR MORE! WHILE YOUR PROGRAMS AREN'T USING SOUND THEN SET THE ALL SOUND OFF FLAG TO 0. IT DEFAULTS TO 1 ON RESET.

These tend to be the two most important equations in converting between Hertz and GB frequency registers:  
(Sounds will have a 2.4% higher frequency on Super GB.)

$$gb = 2048 - (131072 / Hz)$$

$$Hz = 131072 / (2048 - gb)$$

Timer

-----

Sometimes it's useful to have a timer that interrupts at regular intervals for routines that require periodic or percise updates. The timer in the GameBoy has a selectable frequency of 4096, 16384, 65536, or 262144 Hertz. This frequency increments the Timer Counter (TIMA). When it

overflows, it generates an interrupt. It is then loaded with the contents of Timer Modulo (TMA). The following are examples:

;This interval timer interrupts 4096 times per second

```
ld  a,-1
ld  ($FF06),a      ;Set TMA to divide clock by 1
ld  a,4
ld  ($FF07),a      ;Set clock to 4096 Hertz
```

;This interval timer interrupts 65536 times per second

```
ld  a,-4
ld  ($FF06),a      ;Set TMA to divide clock by 4
ld  a,5
ld  ($FF07),a      ;Set clock to 262144 Hertz
```

## Serial I/O

-----

The serial I/O port on the Gameboy is a very simple setup and is crude compared to standard RS-232 (IBM-PC) or RS-485 (Macintosh) serial ports. There are no start or stop bits so the programmer must be more creative when using this port.

During a transfer, a byte is shifted in at the same time that a byte is shifted out. The rate of the shift is deter-

mined by whether the clock source is internal or external. If internal, the bits are shifted out at a rate of 8192Hz (122 microseconds) per bit. The most significant bit is shifted in and out first.

When the internal clock is selected, it drives the clock pin on the game link port and it stays high when not used. During a transfer it will go low eight times to clock in/out each bit.

A programmer initiates a serial transfer by setting bit 7 of \$FF02. This bit may be read and is automatically set to 0 at the completion of transfer. After this bit is set, an interrupt will then occur eight bit clocks later if the serial interrupt is enabled.

If internal clock is selected and serial interrupt is enabled, this interrupt occurs  $122 \times 8$  microseconds later.

If external clock is selected and serial interrupt is enabled, an interrupt will occur eight bit clocks later.

Initiating a serial transfer with external clock will wait forever if no external clock is present. This allows a certain amount of synchronization with each serial port.

The state of the last bit shifted out determines the state of the output line until another transfer takes place.



If a serial transfer with internal clock is performed and no external GameBoy is present, a value of \$FF will be received in the transfer.

The following code causes \$75 to be shifted out the serial port and a byte to be shifted into \$FF01:

```
ld  a,$75
ld  ($FF01),a
ld  a,$81
ld  ($FF02),a
```

#### Interrupt Procedure

-----

The IME (interrupt master enable) flag is reset by DI and prohibits all interrupts. It is set by EI and acknowledges the interrupt setting by the IE register.

1. When an interrupt is generated, the IF flag will be set.
2. If the IME flag is set & the corresponding IE flag is set, the following 3 steps are performed.
3. Reset the IME flag and prevent all interrupts.
4. The PC (program counter) is pushed onto the stack.
5. Jump to the starting address of the interrupt.

Resetting of the IF register, which was the cause of the interrupt, is done by hardware.

During the interrupt, pushing of registers to be used should be performed by the interrupt routine.

Once the interrupt service is in progress, all the interrupts will be prohibited. However, if the IME flag and the IE flag are controlled, a number of interrupt services can be made possible by nesting.

Return from an interrupt routine can be performed by either RETI or RET instruction.

The RETI instruction enables interrupts after doing a return operation.

If a RET is used as the final instruction in an interrupt routine, interrupts will remain disabled unless a EI was used in the interrupt routine or is used at a later time.

The interrupt will be acknowledged during opcode fetch period of each instruction.

## Interrupt Descriptions

-----

The following interrupts only occur if they have been enabled in the Interrupt Enable register (\$FFFF) and

if the interrupts have actually been enabled using the EI instruction.

#### V-Blank -

The V-Blank interrupt occurs ~59.7 times a second on a regular GB and ~61.1 times a second on a Super GB (SGB). This interrupt occurs at the beginning of the V-Blank period. During this period video hardware is not using video ram so it may be freely accessed. This period lasts approximately 1.1 milliseconds.

#### LCDC Status -

There are various reasons for this interrupt to occur as described by the STAT register (\$FF40). One very popular reason is to indicate to the user when the video hardware is about to redraw a given LCD line. This can be useful for dynamically controlling the SCX/SCY registers (\$FF43/\$FF42) to perform special video effects.

#### Timer Overflow -

This interrupt occurs when the TIMA register (\$FF05) changes from \$FF to \$00.

#### Serial Transfer Completion -

This interrupt occurs when a serial transfer has completed on the game link port.

#### High-to-Low of P10-P13 -

This interrupt occurs on a transition of any of the keypad input lines from high to low. Due to the fact that keypad "bounce"\* is virtually always present, software should expect this interrupt to occur one or more times for every button press and one or more times for every button release.

\* - Bounce tends to be a side effect of any button making or breaking a connection. During these periods, it is very common for a small amount of oscillation between high & low states to take place.

#### I/O Registers

-----

##### FF00

Name - P1

Contents - Register for reading joy pad info  
and determining system type. (R/W)

Bit 7 - Not used

Bit 6 - Not used

Bit 5 - P15 out port

Bit 4 - P14 out port

Bit 3 - P13 in port

Bit 2 - P12 in port

Bit 1 - P11 in port

Bit 0 - P10 in port

This is the matrix layout for register \$FF00:

	P14	P15
P10-----	O-Right----	O-A
P11-----	O-Left----	O-B
P12-----	O-Up-----	O-Select
P13-----	O-Down----	O-Start

Example code:

Game: Ms. Pacman

Address: \$3b1

LD A,\$20           <- bit 5 = \$20

LD (\$FF00),A      <- select P14 by setting it low

LD A,(\$FF00)

LD A,(\$FF00)      <- wait a few cycles

```

CPL          <- complement A

AND $0F      <- get only first 4 bits

SWAP A       <- swap it

LD B,A       <- store A in B

LD A,$10

LD ($FF00),A <- select P15 by setting it low

LD A,($FF00)

LD A,($FF00)

LD A,($FF00)

LD A,($FF00)

LD A,($FF00)

LD A,($FF00) <- Wait a few MORE cycles

CPL          <- complement (invert)

AND $0F      <- get first 4 bits

OR B         <- put A and B together


LD B,A       <- store A in D

LD A,($FF8B) <- read old joy data from ram

XOR B        <- toggle w/current button bit

AND B        <- get current button bit back

LD ($FF8C),A <- save in new Joydata storage

LD A,B       <- put original value in A

LD ($FF8B),A <- store it as old joy data


LD A,$30     <- deselect P14 and P15

LD ($FF00),A <- RESET Joypad

RET          <- Return from Subroutine

```

The button values using the above method are such:

\$80 - Start	\$8 - Down
\$40 - Select	\$4 - Up
\$20 - B	\$2 - Left
\$10 - A	\$1 - Right

Let's say we held down A, Start, and Up.

The value returned in accumulator A would be \$94

FF01

Name - SB

Contents - Serial transfer data (R/W)

8 Bits of data to be read/written

FF02

Name - SC

Contents - SIO control (R/W)

Bit 7 - Transfer Start Flag

0: Non transfer

1: Start transfer

Bit 0 - Shift Clock

0: External Clock (500KHz Max.)

1: Internal Clock (8192Hz)

Transfer is initiated by setting the Transfer Start Flag. This bit may be read and is automatically set to 0 at the end of Transfer.

Transmitting and receiving serial data is done simultaneously. The received data is automatically stored in SB.

#### FF04

Name - DIV

Contents - Divider Register (R/W)

This register is incremented 16384 (~16779 on SGB) times a second. Writing any value sets it to \$00.

#### FF05

Name - TIMA

Contents - Timer counter (R/W)

This timer is incremented by a clock frequency specified by the TAC register (\$FF07). The timer generates an interrupt when it overflows.

#### FF06

Name - TMA

Contents - Timer Modulo (R/W)



When the TIMA overflows, this data will be loaded.

FF07

Name - TAC

Contents - Timer Control (R/W)

Bit 2 - Timer Stop

0: Stop Timer

1: Start Timer

Bits 1+0 - Input Clock Select

00: 4.096 KHz (~4.194 KHz SGB)

01: 262.144 KHz (~268.4 KHz SGB)

10: 65.536 KHz (~67.11 KHz SGB)

11: 16.384 KHz (~16.78 KHz SGB)

FF0F

Name - IF

Contents - Interrupt Flag (R/W)

Bit 4: Transition from High to Low of Pin number P10-P13

Bit 3: Serial I/O transfer complete

Bit 2: Timer Overflow

Bit 1: LCDC (see STAT)

Bit 0: V-Blank

The priority and jump address for the above 5 interrupts are:

Interrupt	Priority	Start Address
-----------	----------	---------------

V-Blank	1	\$0040
LCDC Status	2	\$0048 - Modes 0, 1, 2
		LYC=LY coincide (selectable)
Timer Overflow	3	\$0050
Serial Transfer	4	\$0058 - when transfer is complete
Hi-Lo of P10-P13	5	\$0060

\* When more than 1 interrupts occur at the same time only the interrupt with the highest priority can be acknowledged. When an interrupt is used a '0' should be stored in the IF register before the IE register is set.

#### FF10

Name - NR 10

Contents - Sound Mode 1 register, Sweep register (R/W)

Bit 6-4 - Sweep Time

Bit 3 - Sweep Increase/Decrease

0: Addition (frequency increases)

1: Subtraction (frequency decreases)

Bit 2-0 - Number of sweep shift (n: 0-7)

Sweep Time: 000: sweep off - no freq change

001: 7.8 ms (1/128Hz)

010: 15.6 ms (2/128Hz)

011: 23.4 ms (3/128Hz)  
 100: 31.3 ms (4/128Hz)  
 101: 39.1 ms (5/128Hz)  
 110: 46.9 ms (6/128Hz)  
 111: 54.7 ms (7/128Hz)

The change of frequency (NR13,NR14) at each shift  
 is calculated by the following formula where  
 X(0) is initial freq & X(t-1) is last freq:

$$X(t) = X(t-1) \pm X(t-1)/2^n$$

#### FF11

Name - NR 11

Contents - Sound Mode 1 register, Sound length/Wave pattern duty (R/W)

Only Bits 7-6 can be read.

Bit 7-6 - Wave Pattern Duty

Bit 5-0 - Sound length data (t1: 0-63)

Wave Duty: 00: 12.5% ( \_ \_ \_ \_ \_ \_ \_ \_ )  
 01: 25% ( \_ \_ \_ \_ \_ \_ \_ \_ )  
 10: 50% ( \_ \_ \_ \_ \_ \_ \_ \_ ) (default)  
 11: 75% ( \_ \_ \_ \_ \_ \_ \_ \_ )

Sound Length = (64-t1)\*(1/256) seconds

#### FF12

Name - NR 12

Contents - Sound Mode 1 register, Envelope (R/W)

Bit 7-4 - Initial volume of envelope

Bit 3 - Envelope UP/DOWN

0: Attenuate

1: Amplify

Bit 2-0 - Number of envelope sweep (n: 0-7)

(If zero, stop envelope operation.)

Initial volume of envelope is from 0 to \$F.

Zero being no sound.

Length of 1 step =  $n \cdot (1/64)$  seconds

FF13

Name - NR 13

Contents - Sound Mode 1 register, Frequency lo (W)

Lower 8 bits of 11 bit frequency (x).

Next 3 bit are in NR 14 (\$FF14)

FF14

Name - NR 14

Contents - Sound Mode 1 register, Frequency hi (R/W)

Only Bit 6 can be read.

Bit 7 - Initial (when set, sound restarts)

Bit 6 - Counter/consecutive selection

Bit 2-0 - Frequency's higher 3 bits (x)

$$\begin{aligned}\text{Frequency} &= 4194304 / (32 * (2048 - x)) \text{ Hz} \\ &= 131072 / (2048 - x) \text{ Hz}\end{aligned}$$

Counter/consecutive Selection

0 = Regardless of the length data in NR11  
sound can be produced consecutively.

1 = Sound is generated during the time period  
set by the length data in NR11. After this  
period the sound 1 ON flag (bit 0 of NR52)  
is reset.

FF16

Name - NR 21

Contents - Sound Mode 2 register, Sound Length; Wave Pattern Duty (R/W)

Only bits 7-6 can be read.

Bit 7-6 - Wave pattern duty

Bit 5-0 - Sound length data (t1: 0-63)

Wave Duty: 00: 12.5% ( \_ \_ \_ \_ \_ \_ \_ \_ )  
01: 25% ( \_ \_ \_ \_ \_ \_ \_ \_ )  
10: 50% ( \_ \_ \_ \_ \_ \_ \_ \_ ) (default)  
11: 75% ( \_ \_ \_ \_ \_ \_ \_ \_ )

$$\text{Sound Length} = (64 - t1) * (1/256) \text{ seconds}$$

#### FF17

Name - NR 22

Contents - Sound Mode 2 register, envelope (R/W)

Bit 7-4 - Initial volume of envelope

Bit 3 - Envelope UP/DOWN

0: Attenuate

1: Amplify

Bit 2-0 - Number of envelope sweep (n: 0-7)

(If zero, stop envelope operation.)

Initial volume of envelope is from 0 to \$F.

Zero being no sound.

Length of 1 step =  $n \cdot (1/64)$  seconds

#### FF18

Name - NR 23

Contents - Sound Mode 2 register, frequency lo data (W)

Frequency's lower 8 bits of 11 bit data (x).

Next 3 bits are in NR 14 (\$FF19).

#### FF19

Name - NR 24

Contents - Sound Mode 2 register, frequency hi data (R/W)

Only bit 6 can be read.

Bit 7 - Initial (when set, sound restarts)

Bit 6 - Counter/consecutive selection

Bit 2-0 - Frequency's higher 3 bits (x)

$$\begin{aligned}\text{Frequency} &= 4194304 / (32 * (2048 - x)) \text{ Hz} \\ &= 131072 / (2048 - x) \text{ Hz}\end{aligned}$$

Counter/consecutive Selection

0 = Regardless of the length data in NR21  
sound can be produced consecutively.

1 = Sound is generated during the time period  
set by the length data in NR21. After this  
period the sound 2 ON flag (bit 1 of NR52)  
is reset.

FF1A

Name - NR 30

Contents - Sound Mode 3 register, Sound on/off (R/W)

Only bit 7 can be read

Bit 7 - Sound OFF

0: Sound 3 output stop

1: Sound 3 output OK

FF1B

Name - NR 31

Contents - Sound Mode 3 register, sound length (R/W)

Bit 7-0 - Sound length (t1: 0 - 255)

Sound Length =  $(256-t1) * (1/2)$  seconds

FF1C

Name - NR 32

Contents - Sound Mode 3 register, Select output level (R/W)

Only bits 6-5 can be read

Bit 6-5 - Select output level

00: Mute

01: Produce Wave Pattern RAM Data as it is  
(4 bit length)

10: Produce Wave Pattern RAM data shifted once  
to the RIGHT (1/2) (4 bit length)

11: Produce Wave Pattern RAM data shifted twice  
to the RIGHT (1/4) (4 bit length)

\* - Wave Pattern RAM is located from \$FF30-\$FF3f.

FF1D

Name - NR 33

Contents - Sound Mode 3 register, frequency's lower data (W)

Lower 8 bits of an 11 bit frequency (x).



## FF1E

Name - NR 34

Contents - Sound Mode 3 register, frequency's higher data (R/W)

Only bit 6 can be read.

Bit 7 - Initial (when set, sound restarts)

Bit 6 - Counter/consecutive flag

Bit 2-0 - Frequency's higher 3 bits (x).

$$\begin{aligned}\text{Frequency} &= 4194304 / (64 * (2048 - x)) \text{ Hz} \\ &= 65536 / (2048 - x) \text{ Hz}\end{aligned}$$

Counter/consecutive Selection

0 = Regardless of the length data in NR31  
sound can be produced consecutively.

1 = Sound is generated during the time period  
set by the length data in NR31. After this  
period the sound 3 ON flag (bit 2 of NR52)  
is reset.

## FF20

Name - NR 41

Contents - Sound Mode 4 register, sound length (R/W)

Bit 5-0 - Sound length data (t1: 0-63)

$$\text{Sound Length} = (64 - t1) * (1/256) \text{ seconds}$$

## FF21

Name - NR 42

Contents - Sound Mode 4 register, envelope (R/W)

Bit 7-4 - Initial volume of envelope

Bit 3 - Envelope UP/DOWN

0: Attenuate

1: Amplify

Bit 2-0 - Number of envelope sweep (n: 0-7)

(If zero, stop envelope operation.)

Initial volume of envelope is from 0 to \$F.

Zero being no sound.

Length of 1 step =  $n \cdot (1/64)$  seconds

## FF22

Name - NR 43

Contents - Sound Mode 4 register, polynomial counter (R/W)

Bit 7-4 - Selection of the shift clock frequency of the  
polynomial counter

Bit 3 - Selection of the polynomial counter's step

Bit 2-0 - Selection of the dividing ratio of frequencies

Selection of the dividing ratio of frequencies:

000:  $f \cdot 1/2^3 \cdot 2$

001:  $f * 1/2^3 * 1$   
 010:  $f * 1/2^3 * 1/2$   
 011:  $f * 1/2^3 * 1/3$   
 100:  $f * 1/2^3 * 1/4$   
 101:  $f * 1/2^3 * 1/5$   
 110:  $f * 1/2^3 * 1/6$   
 111:  $f * 1/2^3 * 1/7$   $f = 4.194304 \text{ Mhz}$

Selection of the polynomial counter step:

0: 15 steps  
 1: 7 steps

Selection of the shift clock frequency of the polynomial counter:

0000: dividing ratio of frequencies  $* 1/2$   
 0001: dividing ratio of frequencies  $* 1/2^2$   
 0010: dividing ratio of frequencies  $* 1/2^3$   
 0011: dividing ratio of frequencies  $* 1/2^4$   
           :  
           :  
           :  
 0101: dividing ratio of frequencies  $* 1/2^{14}$   
 1110: prohibited code  
 1111: prohibited code

FF23

Name - NR 44

Contents - Sound Mode 4 register, counter/consecutive; initial (R/W)

Only bit 6 can be read.

Bit 7 - Initial (when set, sound restarts)

Bit 6 - Counter/consecutive selection

Counter/consecutive Selection

0 = Regardless of the length data in NR41  
sound can be produced consecutively.

1 = Sound is generated during the time period  
set by the length data in NR41. After this  
period the sound 4 ON flag (bit 3 of NR52)  
is reset.

FF24

Name - NR 50

Contents - Channel control / ON-OFF / Volume (R/W)

Bit 7 - Vin->SO2 ON/OFF

Bit 6-4 - SO2 output level (volume) (# 0-7)

Bit 3 - Vin->SO1 ON/OFF

Bit 2-0 - SO1 output level (volume) (# 0-7)

Vin->SO1 (Vin->SO2)

By synthesizing the sound from sound 1  
through 4, the voice input from Vin  
terminal is put out.

0: no output

1: output OK

#### FF25

Name - NR 51

Contents - Selection of Sound output terminal (R/W)

Bit 7 - Output sound 4 to S02 terminal

Bit 6 - Output sound 3 to S02 terminal

Bit 5 - Output sound 2 to S02 terminal

Bit 4 - Output sound 1 to S02 terminal

Bit 3 - Output sound 4 to S01 terminal

Bit 2 - Output sound 3 to S01 terminal

Bit 1 - Output sound 2 to S01 terminal

Bit 0 - Output sound 1 to S01 terminal

#### FF26

Name - NR 52 (Value at reset: \$F1-GB, \$F0-SGB)

Contents - Sound on/off (R/W)

Bit 7 - All sound on/off

0: stop all sound circuits

1: operate all sound circuits

Bit 3 - Sound 4 ON flag

Bit 2 - Sound 3 ON flag

Bit 1 - Sound 2 ON flag

Bit 0 - Sound 1 ON flag

Bits 0 - 3 of this register are meant to

be status bits to be read. Writing to these bits does NOT enable/disable sound.

If your GB programs don't use sound then write \$00 to this register to save 16% or more on GB power consumption.

#### FF30 - FF3F

Name - Wave Pattern RAM

Contents - Waveform storage for arbitrary sound data

This storage area holds 32 4-bit samples that are played back upper 4 bits first.

#### FF40

Name - LCDC (value \$91 at reset)

Contents - LCD Control (R/W)

Bit 7 - LCD Control Operation \*

0: Stop completely (no picture on screen)

1: operation

Bit 6 - Window Tile Map Display Select

0: \$9800-\$9BFF

1: \$9C00-\$9FFF

Bit 5 - Window Display

0: off

1: on

Bit 4 - BG & Window Tile Data Select

0: \$8800-\$97FF

1: \$8000-\$8FFF <- Same area as OBJ

Bit 3 - BG Tile Map Display Select

0: \$9800-\$9BFF

1: \$9C00-\$9FFF

Bit 2 - OBJ (Sprite) Size

0: 8\*8

1: 8\*16 (width\*height)

Bit 1 - OBJ (Sprite) Display

0: off

1: on

Bit 0 - BG & Window Display

0: off

1: on

\* - Stopping LCD operation (bit 7 from 1 to 0)  
must be performed during V-blank to work  
properly. V-blank can be confirmed when the  
value of LY is greater than or equal to 144.

FF41

Name - STAT

Contents - LCDC Status (R/W)

Bits 6-3 - Interrupt Selection By LCDC Status

Bit 6 - LYC=LY Coincidence (Selectable)

Bit 5 - Mode 10

Bit 4 - Mode 01

Bit 3 - Mode 00

0: Non Selection

1: Selection

Bit 2 - Coincidence Flag

0: LYC not equal to LCDC LY

1: LYC = LCDC LY

Bit 1-0 - Mode Flag

00: During H-Blank

01: During V-Blank

10: During Searching OAM-RAM

11: During Transferring Data to LCD Driver

STAT shows the current status of the LCD controller.

Mode 00: When the flag is 00 it is the H-Blank period  
and the CPU can access the display RAM  
(\$8000-\$9FFF).

Mode 01: When the flag is 01 it is the V-Blank period  
and the CPU can access the display RAM  
(\$8000-\$9FFF).



Mode 10: When the flag is 10 then the OAM is being used (\$FE00-\$FE9F). The CPU cannot access the OAM during this period

Mode 11: When the flag is 11 both the OAM and display RAM are being used. The CPU cannot access either during this period.

The following are typical when the display is enabled:

```

Mode 0  000__000__000__000__000__000__000__
Mode 1  _____11111111111111__
Mode 2  __2__2__2__2__2__2__2__
Mode 3  __33__33__33__33__33__33__3

```

The Mode Flag goes through the values 0, 2, and 3 at a cycle of about 109uS. 0 is present about 48.6uS, 2 about 19uS, and 3 about 41uS. This is interrupted every 16.6ms by the VBlank (1). The mode flag stays set at 1 for about 1.08 ms. (Mode 0 is present between 201-207 clks, 2 about 77-83 clks, and 3 about 169-175 clks. A complete cycle through these states takes 456 clks. VBlank lasts 4560 clks. A complete screen refresh occurs every 70224 clks.)

FF42

Name - SCY

Contents - Scroll Y (R/W)

8 Bit value \$00-\$FF to scroll BG Y screen position.

FF43

Name - SCX

Contents - Scroll X (R/W)

8 Bit value \$00-\$FF to scroll BG X screen position.

FF44

Name - LY

Contents - LCDC Y-Coordinate (R)

The LY indicates the vertical line to which the present data is transferred to the LCD Driver. The LY can take on any value between 0 through 153. The values between 144 and 153 indicate the V-Blank period. Writing will reset the counter.

FF45

Name - LYC

Contents - LY Compare (R/W)

The LYC compares itself with the LY. If the values are the same it causes the STAT to set the coincident flag.

FF46

Name - DMA

Contents - DMA Transfer and Start Address (W)

The DMA Transfer (40\*28 bit) from internal ROM or RAM (\$0000-\$F19F) to the OAM (address \$FE00-\$FE9F) can be performed. It takes 160 microseconds for the transfer.

40\*28 bit = #140 or #\$8C. As you can see, it only transfers \$8C bytes of data. OAM data is \$A0 bytes long, from \$0-\$9F.

But if you examine the OAM data you see that 4 bits are not in use.

40\*32 bit = #\$A0, but since 4 bits for each OAM is not used it's 40\*28 bit.

It transfers all the OAM data to OAM RAM.

The DMA transfer start address can be designated every \$100 from address \$0000-\$F100. That means \$0000, \$0100, \$0200, \$0300....

As can be seen by looking at register \$FF41 Sprite RAM (\$FE00 - \$FE9F) is not always available. A simple routine that many games use to write data to Sprite memory is shown below. Since it copies data to the sprite RAM at the appropriate times it removes that responsibility from the main program.

All of the memory space, except high ram (\$FF80-\$FFFE), is not accessible during DMA. Because of this, the routine below must be copied & executed in high ram. It is usually called from a V-blank Interrupt.

Example program:

```
org $40
jp VBlank

org $ff80
VBlank:
    push af          <- Save A reg & flags
    ld a,BASE_ADRS  <- transfer data from BASE_ADRS
    ld ($ff46),a    <- put A into DMA registers
    ld a,28h        <- loop length
Wait:
    <- We need to wait 160 microseconds.
    dec a           <- 4 cycles - decrease A by 1
    jr nz,Wait      <- 12 cycles - branch if Not Zero to Wait
    pop af          <- Restore A reg & flags
    reti            <- Return from interrupt
```

#### FF47

Name - BGP

Contents - BG & Window Palette Data (R/W)

Bit 7-6 - Data for Dot Data 11 (Normally darkest color)

Bit 5-4 - Data for Dot Data 10

Bit 3-2 - Data for Dot Data 01

Bit 1-0 - Data for Dot Data 00 (Normally lightest color)

This selects the shade of grays to use for the background (BG) & window pixels. Since each pixel uses 2 bits, the corresponding shade will be selected from here.

#### FF48

Name - OBP0

Contents - Object Palette 0 Data (R/W)

This selects the colors for sprite palette 0.

It works exactly as BGP (\$FF47) except each value of 0 is transparent.

#### FF49

Name - OBP1

Contents - Object Palette 1 Data (R/W)

This Selects the colors for sprite palette 1.

It works exactly as OBP0 (\$FF48).

See BGP for details.

#### FF4A

Name - WY

Contents - Window Y Position (R/W)

$0 \leq WY \leq 143$

WY must be greater than or equal to 0 and must be less than or equal to 143 for window to be visible.

#### FF4B

Name - WX

Contents - Window X Position (R/W)

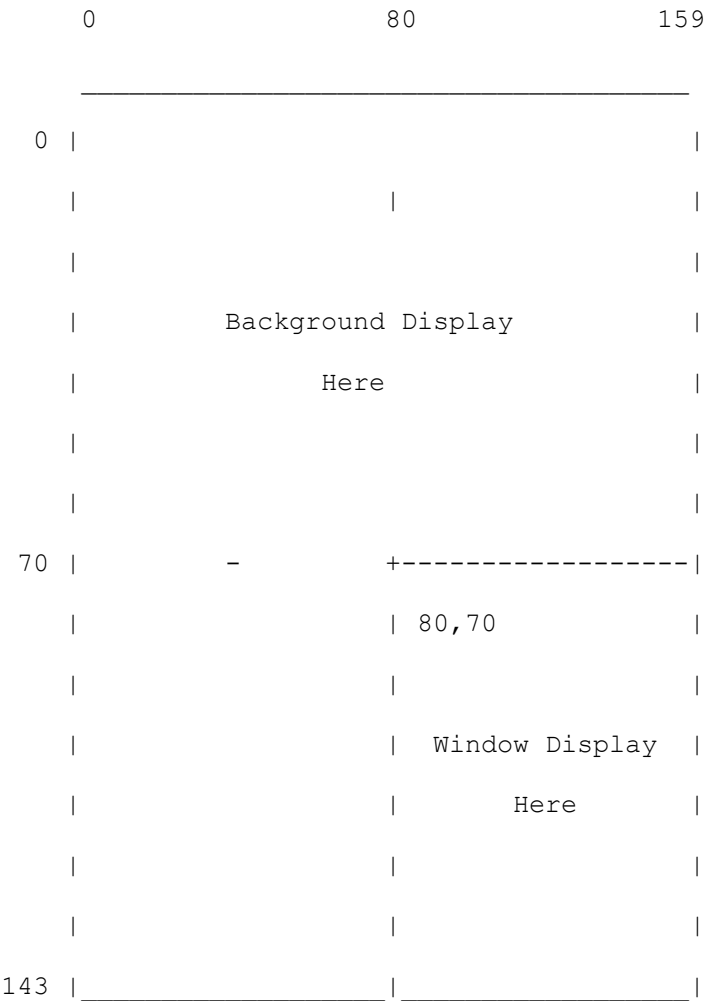
$0 \leq WX \leq 166$

WX must be greater than or equal to 0 and must be less than or equal to 166 for window to be visible.

WX is offset from absolute screen coordinates by 7. Setting the window to WX=7, WY=0 will put the upper left corner of the window at absolute screen coordinates 0,0.

Lets say WY = 70 and WX = 87.

The window would be positioned as so:



OBJ Characters (Sprites) can still enter the window. None of the window colors are transparent so any background tiles under the window are hidden.

FFFF

Name - IE

Contents - Interrupt Enable (R/W)

Bit 4: Transition from High to Low of Pin  
number P10-P13.

Bit 3: Serial I/O transfer complete

Bit 2: Timer Overflow

Bit 1: LCDC (see STAT)

Bit 0: V-Blank

0: disable

1: enable