Pan Docs

About the Pan Docs

```
================================================================
      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
        CPU, SGB, CGB, AUX specs by Martin Korth


            Last updated 10/2001 by nocash
          Previously updated 4-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.

Available Document Formats

The present version of this document is available in Text and Html format:

```
http://www.work.de/nocash/pandocs.txt
http://www.work.de/nocash/pandocs.htm
```

Also, a copy of this document is included in the manual of newer versions of the no$gmb debugger, because of recent piracy attacks (many thanks and best wishes go to hell) I have currently no intention to publish any such or further no$gmb updates though.

Game Boy Technical Data

```
CPU          - 8-bit (Similar to the Z80 processor)
Clock Speed  - 4.194304MHz (4.295454MHz for SGB, max. 8.4MHz for CGB)
Work RAM     - 8K Byte (32K Byte for CGB)
Video RAM    - 8K Byte (16K Byte for CGB)
Screen Size  - 2.6"
Resolution   - 160x144 (20x18 tiles)
Max sprites  - Max 40 per screen, 10 per line
Sprite sizes - 8x8 or 8x16
Palettes     - 1x4 BG, 2x3 OBJ (for CGB: 8x4 BG, 8x3 OBJ)
Colors       - 4 grayshades (32768 colors for CGB)
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, DC3V 0.6W for CGB)
```

Memory Map

The gameboy is having a 16bit address bus, that is used to address ROM, RAM, and I/O registers.

General Memory Map

```
0000-3FFF   16KB ROM Bank 00     (in cartridge, fixed at bank 00)
4000-7FFF   16KB ROM Bank 01..NN (in cartridge, switchable bank number)
8000-9FFF   8KB Video RAM (VRAM) (switchable bank 0-1 in CGB Mode)
A000-BFFF   8KB External RAM     (in cartridge, switchable bank, if any)
C000-CFFF   4KB Work RAM Bank 0 (WRAM)
D000-DFFF   4KB Work RAM Bank 1 (WRAM)  (switchable bank 1-7 in CGB Mode)
E000-FDFF   Same as C000-DDFF (ECHO)    (typically not used)
FE00-FE9F   Sprite Attribute Table (OAM)
FEA0-FEFF   Not Usable
FF00-FF7F   I/O Ports
FF80-FFFE   High RAM (HRAM)
FFFF        Interrupt Enable Register
```

Jump Vectors in First ROM Bank

The following addresses are supposed to be used as jump vectors:

```
0000,0008,0010,0018,0020,0028,0030,0038   for RST commands
0040,0048,0050,0058,0060                  for Interrupts
```

However, the memory may be used for any other purpose in case that your program doesn't use any (or only some) RST commands or Interrupts. RST commands are 1-byte opcodes that work similiar to CALL opcodes, except that the destination address is fixed.

Cartridge Header in First ROM Bank

The memory at 0100-014F contains the cartridge header. This area contains information about the program, its entry point, checksums, information about the used MBC chip, the ROM and RAM sizes, etc. Most of the bytes in this area are required to be specified correctly. For more information read the chapter about The Cartridge Header.

External Memory and Hardware

The areas from 0000-7FFF and A000-BFFF may be used to connect external hardware. The first area is typically used to address ROM (read only, of course), cartridges with Memory Bank Controllers (MBCs) are additionally using this area to output data (write only) to the MBC chip. The second area is often used to address external RAM, or to address other external hardware (Real Time Clock, etc). External memory is often battery buffered, and may hold saved game positions and high scrore tables (etc.) even when the gameboy is turned of, or when the cartridge is removed. For specific information read the chapter about Memory Bank Controllers.

Video Display

Video I/O Registers

Video Memory

LCD Control Register

FF40 - LCDC - LCD Control (R/W)

```
Bit 7 - LCD Display Enable             (0=Off, 1=On)
Bit 6 - Window Tile Map Display Select (0=9800-9BFF, 1=9C00-9FFF)
Bit 5 - Window Display Enable          (0=Off, 1=On)
Bit 4 - BG & Window Tile Data Select   (0=8800-97FF, 1=8000-8FFF)
Bit 3 - BG Tile Map Display Select     (0=9800-9BFF, 1=9C00-9FFF)
Bit 2 - OBJ (Sprite) Size              (0=8x8, 1=8x16)
Bit 1 - OBJ (Sprite) Display Enable    (0=Off, 1=On)
Bit 0 - BG Display (for CGB see below) (0=Off, 1=On)
```

LCDC.7 - LCD Display Enable

CAUTION: Stopping LCD operation (Bit 7 from 1 to 0) may be performed during V-Blank ONLY, disabeling the display outside of the V-Blank period may damage the hardware. This appears to be a serious issue, Nintendo is reported to reject any games that do not follow this rule.
V-blank can be confirmed when the value of LY is greater than or equal to 144. When the display is disabled the screen is blank (white), and VRAM and OAM can be accessed freely.

--- LCDC.0 has different Meanings depending on Gameboy Type ---

LCDC.0 - 1) Monochrome Gameboy and SGB: BG Display

When Bit 0 is cleared, the background becomes blank (white). Window and Sprites may still be displayed (if enabled in Bit 1 and/or Bit 5).

LCDC.0 - 2) CGB in CGB Mode: BG and Window Master Priority

When Bit 0 is cleared, the background and window lose their priority - the sprites will be always displayed on top of background and window, independently of the priority flags in OAM and BG Map attributes.

LCDC.0 - 3) CGB in Non CGB Mode: BG and Window Display
When Bit 0 is cleared, both background and window become blank (white), ie. the Window Display Bit (Bit 5) is ignored in that case. Only Sprites may still be displayed (if enabled in Bit 1).
This is a possible compatibility problem - any monochrome games (if any) that disable the background, but still want to display the window wouldn't work properly on CGBs.

LCD Status Register

FF41 - STAT - LCDC Status (R/W)

```
Bit 6 - LYC=LY Coincidence Interrupt (1=Enable) (Read/Write)
Bit 5 - Mode 2 OAM Interrupt         (1=Enable) (Read/Write)
Bit 4 - Mode 1 V-Blank Interrupt     (1=Enable) (Read/Write)
Bit 3 - Mode 0 H-Blank Interrupt     (1=Enable) (Read/Write)
Bit 2 - Coincidence Flag  (0:LYC<>LY, 1:LYC=LY) (Read Only)
Bit 1-0 - Mode Flag       (Mode 0-3, see below) (Read Only)
          0: During H-Blank
          1: During V-Blank
          2: During Searching OAM-RAM
          3: During Transfering Data to LCD Driver
```

The two lower STAT bits show the current status of the LCD controller.

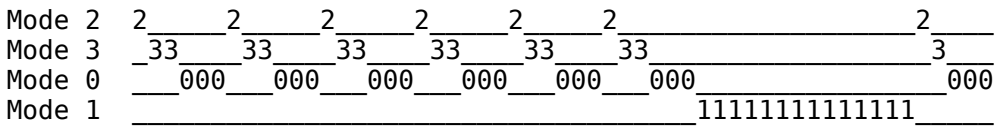
```
Mode 0: The LCD controller is in the H-Blank period and
        the CPU can access both the display RAM (8000h-9FFFh)
        and OAM (FE00h-FE9Fh)


Mode 1: The LCD contoller is in the V-Blank period (or the
        display is disabled) and the CPU can access both the
        display RAM (8000h-9FFFh) and OAM (FE00h-FE9Fh)


Mode 2: The LCD controller is reading from OAM memory.
        The CPU <cannot> access OAM memory (FE00h-FE9Fh)
        during this period.


Mode 3: The LCD controller is reading from both OAM and VRAM,
        The CPU <cannot> access OAM and VRAM during this period.
        CGB Mode: Cannot access Palette Data (FF69,FF6B) either.
```

The following are typical when the display is enabled:



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.)

LCD Interrupts

INT 40 - V-Blank Interrupt
The V-Blank interrupt occurs ca. 59.7 times a second on a regular GB and ca. 61.1 times a second on a Super GB (SGB). This interrupt occurs at the beginning of the V-Blank period (LY=144).
During this period video hardware is not using video ram so it may be freely accessed. This period lasts approximately 1.1 milliseconds.

INT 48 - LCDC Status Interrupt
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.

LCD Position and Scrolling

FF42 - SCY - Scroll Y (R/W)
FF43 - SCX - Scroll X (R/W)
Specifies the position in the 256x256 pixels BG map (32x32 tiles) which is to be displayed at the upper/left LCD display position.
Values in range from 0-255 may be used for X/Y each, the video controller automatically wraps back to the upper (left) position in BG map when drawing exceeds the lower (right) border of the BG map area.

FF44 - LY - 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 - LYC - LY Compare (R/W)
The gameboy permanently compares the value of the LYC and LY registers. When both values are identical, the coincident bit in the STAT register

becomes set, and (if enabled) a STAT interrupt is requested.

FF4A - WY - Window Y Position (R/W)
FF4B - WX - Window X Position minus 7 (R/W)
Specifies the upper/left positions of the Window area. (The window is an alternate background area which can be displayed above of the normal background. OBJs (sprites) may be still displayed above or behinf the window, just as for normal BG.)
The window becomes visible (if enabled) when positions are set in range WX=0..166, WY=0..143. A postion of WX=7, WY=0 locates the window at upper left, it is then completly covering normal background.

LCD Monochrome Palettes

FF47 - BGP - BG Palette Data (R/W) - Non CGB Mode Only
This register assigns gray shades to the color numbers of the BG and Window tiles.

```
Bit 7-6 - Shade for Color Number 3
Bit 5-4 - Shade for Color Number 2
Bit 3-2 - Shade for Color Number 1
Bit 1-0 - Shade for Color Number 0
```

The four possible gray shades are:

```
0  White
1  Light gray
2  Dark gray
3  Black
```

In CGB Mode the Color Palettes are taken from CGB Palette Memory instead.

FF48 - OBP0 - Object Palette 0 Data (R/W) - Non CGB Mode Only
This register assigns gray shades for sprite palette 0. It works exactly as BGP (FF47), except that the lower two bits aren't used because sprite data 00 is transparent.

FF49 - OBP1 - Object Palette 1 Data (R/W) - Non CGB Mode Only
This register assigns gray shades for sprite palette 1. It works exactly as BGP (FF47), except that the lower two bits aren't used because sprite data 00 is transparent.

LCD Color Palettes (CGB only)

FF68 - BCPS/BGPI - CGB Mode Only - Background Palette Index
This register is used to address a byte in the CGBs Background Palette Memory. Each two byte in that memory define a color value. The first 8 bytes define Color 0-3 of Palette 0 (BGP0), and so on for BGP1-7.

```
Bit 0-5   Index (00-3F)
```

```
Bit 7     Auto Increment  (0=Disabled, 1=Increment after Writing)
```

Data can be read/written to/from the specified index address through Register FF69. When the Auto Increment Bit is set then the index is automatically incremented after each <write> to FF69. Auto Increment has no effect when <reading> from FF69, so the index must be manually incremented in that case.

FF69 - BCPD/BGPD - CGB Mode Only - Background Palette Data

This register allows to read/write data to the CGBs Background Palette Memory, addressed through Register FF68.
Each color is defined by two bytes (Bit 0-7 in first byte).

```
Bit 0-4   Red Intensity   (00-1F)
Bit 5-9   Green Intensity (00-1F)
Bit 10-14 Blue Intensity  (00-1F)
```

Much like VRAM, Data in Palette Memory cannot be read/written during the time when the LCD Controller is reading from it. (That is when the STAT register indicates Mode 3).
Note: Initially all background colors are initialized as white.

FF6A - OCPS/OBPI - CGB Mode Only - Sprite Palette Index
FF6B - OCPD/OBPD - CGB Mode Only - Sprite Palette Data

These registers are used to initialize the Sprite Palettes OBP0-7, identically as described above for Background Palettes. Note that four colors may be defined for each OBP Palettes - but only Color 1-3 of each Sprite Palette can be displayed, Color 0 is always transparent, and can be initialized to a don't care value.
Note: Initially all sprite colors are uninitialized.

RGB Translation by CGBs

When developing graphics on PCs, note that the RGB values will have different appearance on CGB displays as on VGA monitors:
The highest intensity will produce Light Gray color rather than White. The intensities are not linear; the values 10h-1Fh will all appear very bright, while medium and darker colors are ranged at 00h-0Fh.
The CGB display will mix colors quite oddly, increasing intensity of only one R,G,B color will also influence the other two R,G,B colors.
For example, a color setting of 03EFh (Blue=0, Green=1Fh, Red=0Fh) will appear as Neon Green on VGA displays, but on the CGB it'll produce a decently washed out Yellow.

RGB Translation by GBAs

Even though GBA is described to be compatible to CGB games, most CGB games are completely unplayable on GBAs because most colors are invisible (black). Of course, colors such like Black and White will appear the same on both CGB and GBA, but medium intensities are arranged completely different.
Intensities in range 00h..0Fh are invisible/black (unless eventually under best sunlight circumstances, and when gazing at the screen under obscure viewing angles), unfortunately, these intensities are regulary used by most existing CGB games for medium and darker colors.
Newer CGB games may avoid this effect by changing palette data when detecting GBA hardware. A relative simple method would be using the formula GBA=CGB/2+10h for each R,G,B intensity, probably the result won't be perfect, and (once colors became visible) it may turn out that the color mixing is different also, anyways, it'd be still ways better than no conversion.
Asides, this translation method should have been VERY easy to implement in GBA hardware directly, even though Nintendo obviously failed to do so. How did they say, This seal is your assurance for excellence in workmanship and so on?

LCD VRAM Bank (CGB only)

FF4F - VBK - CGB Mode Only - VRAM Bank
This 1bit register selects the current Video Memory (VRAM) Bank.

```
Bit 0 - VRAM Bank (0-1)
```

Bank 0 contains 192 Tiles, and two background maps, just as for monochrome games. Bank 1 contains another 192 Tiles, and color attribute maps for the background maps in bank 0.

LCD OAM DMA Transfers

FF46 - DMA - DMA Transfer and Start Address (W)
Writing to this register launches a DMA transfer from ROM or RAM to OAM memory (sprite attribute table). The written value specifies the transfer source address divided by 100h, ie. source & destination are:

```
Source:      XX00-XX9F   ;XX in range from 00-F1h
Destination: FE00-FE9F
```

It takes 160 microseconds until the transfer has completed (80 microseconds in CGB Double Speed Mode), during this time the CPU can access only HRAM (memory at FF80-FFFE). For this reason, the programmer must copy a short procedure into HRAM, and use this procedure to start the transfer from inside HRAM, and wait until the transfer has finished:

```
   ld  (0FF46h),a ;start DMA transfer, a=start address/100h
   ld  a,28h      ;delay...
  wait:           ;total 5x40 cycles, approx 200ms
   dec a          ;1 cycle
   jr  nz,wait    ;4 cycles
```

Most programs are executing this procedure from inside of their VBlank procedure, but it is possible to execute it during display redraw also, allowing to display more than 40 sprites on the screen (ie. for example 40 sprites in upper half, and other 40 sprites in lower half of the screen).

LCD VRAM DMA Transfers (CGB only)

FF51 - HDMA1 - CGB Mode Only - New DMA Source, High
FF52 - HDMA2 - CGB Mode Only - New DMA Source, Low
FF53 - HDMA3 - CGB Mode Only - New DMA Destination, High
FF54 - HDMA4 - CGB Mode Only - New DMA Destination, Low
FF55 - HDMA5 - CGB Mode Only - New DMA Length/Mode/Start
These registers are used to initiate a DMA transfer from ROM or RAM to VRAM. The Source Start Address may be located at 0000-7FF0 or A000-

DFF0, the lower four bits of the address are ignored (treated as zero). The Destination Start Address may be located at 8000-9FF0, the lower four bits of the address are ignored (treated as zero), the upper 3 bits are ignored either (destination is always in VRAM).

Writing to FF55 starts the transfer, the lower 7 bits of FF55 specify the Transfer Length (divided by 10h, minus 1). Ie. lengths of 10h-800h bytes can be defined by the values 00h-7Fh. And the upper bit of FF55 indicates the Transfer Mode:

Bit7=0 - General Purpose DMA
When using this transfer method, all data is transferred at once. The execution of the program is halted until the transfer has completed. Note that the General Purpose DMA blindly attempts to copy the data, even if the LCD controller is currently accessing VRAM. So General Purpose DMA should be used only if the Display is disabled, or during V-Blank, or (for rather short blocks) during H-Blank.
The execution of the program continues when the transfer has been completed, and FF55 then contains a value if FFh.

Bit7=1 - H-Blank DMA
The H-Blank DMA transfers 10h bytes of data during each H-Blank, ie. at LY=0-143, no data is transferred during V-Blank (LY=144-153), but the transfer will then continue at LY=00. The execution of the program is halted during the separate transfers, but the program execution continues during the 'spaces' between each data block.
Note that the program may not change the Destination VRAM bank (FF4F), or the Source ROM/RAM bank (in case data is transferred from bankable memory) until the transfer has completed!
Reading from Register FF55 returns the remaining length (divided by 10h, minus 1), a value of 0FFh indicates that the transfer has completed. It is also possible to terminate an active H-Blank transfer by writing zero to Bit 7 of FF55. In that case reading from FF55 may return any value for the lower 7 bits, but Bit 7 will be read as "1".

Confirming if the DMA Transfer is Active
Reading Bit 7 of FF55 can be used to confirm if the DMA transfer is active (1=Not Active, 0=Active). This works under any circumstances - after completion of General Purpose, or H-Blank Transfer, and after manually terminating a H-Blank Transfer.

Transfer Timings
In both Normal Speed and Double Speed Mode it takes about 8us to transfer a block of 10h bytes. That are 8 cycles in Normal Speed Mode, and 16 'fast' cycles in Double Speed Mode.
Older MBC controllers (like MBC1-4) and slower ROMs are not guaranteed to support General Purpose or H-Blank DMA, that's because there are always 2 bytes transferred per microsecond (even if the itself program runs it Normal Speed Mode).

VRAM Tile Data

Tile Data is stored in VRAM at addresses 8000h-97FFh, this area defines the Bitmaps for 192 Tiles. In CGB Mode 384 Tiles can be defined, because memory at 0:8000h-97FFh and at 1:8000h-97FFh is used.

Each tile is sized 8x8 pixels and has a color depth of 4 colors/gray shades. Tiles can be displayed as part of the Background/Window map, and/or as OAM tiles (foreground sprites). Note that foreground sprites may have only 3 colors, because color 0 is transparent.

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 and the background. 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.

Each Tile occupies 16 bytes, where each 2 bytes represent a line:

```
Byte 0-1  First Line (Upper 8 pixels)
Byte 2-3  Next Line
etc.
```

For each line, the first byte defines the least significant bits of the color numbers for each pixel, and the second byte defines the upper bits of the color numbers. In either case, Bit 7 is the leftmost pixel, and Bit 0 the rightmost.

So, each pixel is having a color number in range from 0-3. The color numbers are translated into real colors (or gray shades) depending on the current palettes. The palettes are defined through registers FF47-FF49 (Non CGB Mode), and FF68-FF6B (CGB Mode).

VRAM Background Maps

The gameboy contains two 32x32 tile background maps in VRAM at addresses 9800h-9BFFh and 9C00h-9FFFh. Each can be used either to display "normal" background, or "window" background.

BG Map Tile Numbers
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 via LCDC register.

BG Map Attributes (CGB Mode only)
In CGB Mode, an additional map of 32x32 bytes is stored in VRAM Bank 1 (each byte defines attributes for the corresponding tile-number map entry in VRAM Bank 0):

```
Bit 0-2  Background Palette number  (BGP0-7)
Bit 3    Tile VRAM Bank number      (0=Bank 0, 1=Bank 1)
Bit 4    Not used
Bit 5    Horizontal Flip            (0=Normal, 1=Mirror horizontally)
Bit 6    Vertical Flip              (0=Normal, 1=Mirror vertically)
Bit 7    BG-to-OAM Priority         (0=Use OAM priority bit, 1=BG Priority)
```

When Bit 7 is set, the corresponding BG tile will have priority above all OBJs (regardless of the priority bits in OAM memory). There's also an Master Priority flag in LCDC register Bit 0 which overrides all other priority bits when cleared.

As one background tile has a size of 8x8 pixels, the BG maps may hold a picture of 256x256 pixels, an area of 160x144 pixels of this picture can be displayed on the LCD screen.

Normal Background (BG)
The SCY and SCX registers can be used to scroll the background, allowing to select the origin of the visible 160x144 pixel area within the total 256x256 pixel background map. Background wraps around the screen (i.e. when part of it goes off the screen, it appears on the opposite side.)

The Window
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 WX and WY registers. Screen coordinates of the top left corner of a window are WX-7,WY. The tiles for the window are stored in the Tile Data Table. 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.

VRAM Sprite Attribute Table (OAM)

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 BG 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. Each of the 40 entries consists of four bytes with the following meanings:

Byte0 - Y Position
Specifies the sprites vertical position on the screen (minus 16).
An offscreen value (for example, Y=0 or Y>=160) hides the sprite.

Byte1 - X Position
Specifies the sprites horizontal position on the screen (minus 8).
An offscreen value (X=0 or X>=168) hides the sprite, but the sprite
still affects the priority ordering - a better way to hide a sprite is to set its Y-coordinate offscreen.

Byte2 - Tile/Pattern Number
Specifies the sprites Tile Number (00-FF). This (unsigned) value selects a tile from memory at 8000h-8FFFh. In CGB Mode this could be either in VRAM Bank 0 or 1, depending on Bit 3 of the following byte.
In 8x16 mode, the lower bit of the tile number is ignored. Ie. the upper 8x8 tile is "NN AND FEh", and the lower 8x8 tile is "NN OR 01h".

Byte3 - Attributes/Flags:

```
  Bit7   OBJ-to-BG Priority (0=OBJ Above BG, 1=OBJ Behind BG color 1-3)
         (Used for both BG and Window. BG color 0 is always behind OBJ)
  Bit6   Y flip          (0=Normal, 1=Vertically mirrored)
```

```
Bit5   X flip          (0=Normal, 1=Horizontally mirrored)
Bit4   Palette number  **Non CGB Mode Only** (0=OBP0, 1=OBP1)
Bit3   Tile VRAM-Bank  **CGB Mode Only**     (0=Bank 0, 1=Bank 1)
Bit2-0 Palette number  **CGB Mode Only**     (OBP0-7)
```

Sprite Priorities and Conflicts
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. This applies in Non CGB Mode only.
When sprites with the same x coordinate values overlap, they have priority according to table ordering. (i.e. $FE00 - highest, $FE04 - next highest, etc.) In CGB Mode priorities are always assigned like this.

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.

Writing Data to OAM Memory
The recommened method is to write the data to normal RAM first, and to copy that RAM to OAM by using the DMA transfer function, initiated through DMA register (FF46).
Beside for that, it is also possible to write data directly to the OAM area by using normal LD commands, this works only during the H-Blank and V-Blank periods. The current state of the LCD controller can be read out from the STAT register (FF41).

Accessing VRAM and OAM

CAUTION
When the LCD Controller is drawing the screen it is directly reading from Video Memory (VRAM) and from the Sprite Attribute Table (OAM). During these periods the Gameboy CPU may not access the VRAM and OAM. That means, any attempts to write to VRAM/OAM are ignored (the data remains unchanged). And any attempts to read from VRAM/OAM will return undefined data (typically a value of FFh).

For this reason the program should verify if VRAM/OAM is accessable before actually reading or writing to it. This is usually done by reading the Mode Bits from the STAT Register (FF41). When doing this (as described in the examples below) you should take care that no interrupts occur between the wait loops and the following memory access - the memory is guaranted to be accessable only for a few cycles directly after the wait loops have completed.

VRAM (memory at 8000h-9FFFh) is accessable during Mode 0-2

```
Mode 0 - H-Blank Period,
Mode 1 - V-Blank Period, and
Mode 2 - Searching OAM Period
```

A typical procedure that waits for accessibility of VRAM would be:

```
  ld   hl,0FF41h    ;-STAT Register
@@wait:             ;\
```

```
  bit  1,(hl)       ; Wait until Mode is 0 or 1
  jr   nz,@@wait    ;/
```

Even if the procedure gets executed at the <end> of Mode 0 or 1, it is still proof to assume that VRAM can be accessed for a few more cycles because in either case the following period is Mode 2 which allows access to VRAM either.
In CGB Mode an alternate method to write data to VRAM is to use the HDMA Function (FF51-FF55).

OAM (memory at FE00h-FE9Fh) is accessable during Mode 0-1

```
  Mode 0 - H-Blank Period, and
  Mode 1 - V-Blank Period
```

Beside for that, OAM can be accessed at any time by using the DMA Function (FF46). When directly reading or writing to OAM, a typical procedure that waits for accessibilty or OAM Memory would be:

```
  ld   hl,0FF41h    ;-STAT Register
 @@wait1:           ;\
  bit  1,(hl)       ; Wait until Mode is -NOT- 0 or 1
  jr   z,@@wait1    ;/
 @@wait2:           ;\
  bit  1,(hl)       ; Wait until Mode 0 or 1 -BEGINS-
  jr   nz,@@wait2   ;/
```

The two wait loops ensure that Mode 0 or 1 will last for a few clock cycles after completion of the procedure. In V-Blank period it might be recommended to skip the whole procedure - and in most cases using the above mentioned DMA function would be more recommended anyways.

Note
When the display is disabled, both VRAM and OAM are accessable at any time. The downside is that the screen is blank (white) during this period, so that disabling the display would be recommended only during initialization.

Sound Controller

Sound Overview

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.

(Sounds will have a 2.4% higher frequency on Super GB.)

Sound Channel 1 - Tone & Sweep

FF10 - NR10 - Channel 1 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) +/- X(t-1)/2^n
```

FF11 - NR11 - Channel 1 Sound length/Wave pattern duty (R/W)

```
Bit 7-6 - Wave Pattern Duty (Read/Write)
Bit 5-0 - Sound length data (Write Only) (t1: 0-63)
```

Wave Duty:

```
00: 12.5% ( _-------_-------_------- )
01: 25%   ( __------__------__------ )
```

```
10: 50%   ( ____----____----____---- ) (normal)
11: 75%   ( ______--______--______-- )
```

Sound Length = (64-t1)*(1/256) seconds
The Length value is used only if Bit 6 in NR14 is set.

FF12 - NR12 - Channel 1 Volume Envelope (R/W)

```
Bit 7-4 - Initial Volume of envelope (0-0Fh) (0=No Sound)
Bit 3   - Envelope Direction (0=Decrease, 1=Increase)
Bit 2-0 - Number of envelope sweep (n: 0-7)
          (If zero, stop envelope operation.)
```

Length of 1 step = n*(1/64) seconds

FF13 - NR13 - Channel 1 Frequency lo (Write Only)

Lower 8 bits of 11 bit frequency (x).
Next 3 bit are in NR14 ($FF14)

FF14 - NR14 - Channel 1 Frequency hi (R/W)

```
Bit 7   - Initial (1=Restart Sound)     (Write Only)
Bit 6   - Counter/consecutive selection (Read/Write)
          (1=Stop output when length in NR11 expires)
Bit 2-0 - Frequency's higher 3 bits (x) (Write Only)
```

Frequency = 131072/(2048-x) Hz

Sound Channel 2 - Tone

This sound channel works exactly as channel 1, except that it doesn't have a Tone Envelope/Sweep Register.

FF16 - NR21 - Channel 2 Sound Length/Wave Pattern Duty (R/W)

```
Bit 7-6 - Wave Pattern Duty (Read/Write)
Bit 5-0 - Sound length data (Write Only) (t1: 0-63)
```

Wave Duty:

```
00: 12.5% ( _-------_-------_------- )
01: 25%   ( __------__------__------ )
10: 50%   ( ____----____----____---- ) (normal)
11: 75%   ( ______--______--______-- )
```

Sound Length = (64-t1)*(1/256) seconds
The Length value is used only if Bit 6 in NR24 is set.

FF17 - NR22 - Channel 2 Volume Envelope (R/W)

```
Bit 7-4 - Initial Volume of envelope (0-0Fh) (0=No Sound)
Bit 3   - Envelope Direction (0=Decrease, 1=Increase)
Bit 2-0 - Number of envelope sweep (n: 0-7)
          (If zero, stop envelope operation.)
```

Length of 1 step = n*(1/64) seconds

FF18 - NR23 - Channel 2 Frequency lo data (W)
Frequency's lower 8 bits of 11 bit data (x).
Next 3 bits are in NR24 ($FF19).

FF19 - NR24 - Channel 2 Frequency hi data (R/W)

```
Bit 7   - Initial (1=Restart Sound)     (Write Only)
Bit 6   - Counter/consecutive selection (Read/Write)
          (1=Stop output when length in NR21 expires)
Bit 2-0 - Frequency's higher 3 bits (x) (Write Only)
```

Frequency = 131072/(2048-x) Hz

Sound Channel 3 - Wave Output

This channel can be used to output digital sound, the length of the sample buffer (Wave RAM) is limited to 32 digits. This sound channel can be also used to output normal tones when initializing the Wave RAM by a square wave. This channel doesn't have a volume envelope register.

FF1A - NR30 - Channel 3 Sound on/off (R/W)

```
Bit 7 - Sound Channel 3 Off  (0=Stop, 1=Playback)  (Read/Write)
```

FF1B - NR31 - Channel 3 Sound Length

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

Sound Length = (256-t1)*(1/256) seconds
This value is used only if Bit 6 in NR34 is set.

FF1C - NR32 - Channel 3 Select output level (R/W)

```
Bit 6-5 - Select output level (Read/Write)
```

Possible Output levels are:

```
0: Mute (No sound)
1: 100% Volume (Produce Wave Pattern RAM Data as it is)
2:  50% Volume (Produce Wave Pattern RAM data shifted once to the right)
3:  25% Volume (Produce Wave Pattern RAM data shifted twice to the right)
```

FF1D - NR33 - Channel 3 Frequency's lower data (W)

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

FF1E - NR34 - Channel 3 Frequency's higher data (R/W)

```
  Bit 7   - Initial (1=Restart Sound)     (Write Only)
  Bit 6   - Counter/consecutive selection (Read/Write)
            (1=Stop output when length in NR31 expires)
  Bit 2-0 - Frequency's higher 3 bits (x) (Write Only)
```

Frequency = 4194304/(64*(2048-x)) Hz = 65536/(2048-x) Hz

FF30-FF3F - 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.

Sound Channel 4 - Noise

This channel is used to output white noise. This is done by randomly switching the amplitude between high and low at a given frequency. Depending on the frequency the noise will appear 'harder' or 'softer'.

It is also possible to influence the function of the random generator, so the that the output becomes more regular, resulting in a limited ability to output Tone instead of Noise.

FF20 - NR41 - Channel 4 Sound Length (R/W)

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

Sound Length = (64-t1)*(1/256) seconds
The Length value is used only if Bit 6 in NR44 is set.

FF21 - NR42 - Channel 4 Volume Envelope (R/W)

```
  Bit 7-4 - Initial Volume of envelope (0-0Fh) (0=No Sound)
  Bit 3   - Envelope Direction (0=Decrease, 1=Increase)
  Bit 2-0 - Number of envelope sweep (n: 0-7)
            (If zero, stop envelope operation.)
```

Length of 1 step = n*(1/64) seconds

FF22 - NR43 - Channel 4 Polynomial Counter (R/W)

The amplitude is randomly switched between high and low at the given frequency. A higher frequency will make the noise to appear 'softer'.
When Bit 3 is set, the output will become more regular, and some frequencies will sound more like Tone than Noise.

```
  Bit 7-4 - Shift Clock Frequency (s)
  Bit 3   - Counter Step/Width (0=15 bits, 1=7 bits)
  Bit 2-0 - Dividing Ratio of Frequencies (r)
```

Frequency = 524288 Hz / r / 2^(s+1) ;For r=0 assume r=0.5 instead

FF23 - NR44 - Channel 4 Counter/consecutive; Inital (R/W)

```
Bit 7   - Initial (1=Restart Sound)     (Write Only)
Bit 6   - Counter/consecutive selection (Read/Write)
          (1=Stop output when length in NR41 expires)
```

Sound Control Registers

FF24 - NR50 - Channel control / ON-OFF / Volume (R/W)

The volume bits specify the "Master Volume" for Left/Right sound output.

```
Bit 7   - Output Vin to SO2 terminal (1=Enable)
Bit 6-4 - SO2 output level (volume)  (0-7)
Bit 3   - Output Vin to SO1 terminal (1=Enable)
Bit 2-0 - SO1 output level (volume)  (0-7)
```

The Vin signal is received from the game cartridge bus, allowing external hardware in the cartridge to supply a fifth sound channel, additionally to the gameboys internal four channels. As far as I know this feature isn't used by any existing games.

FF25 - NR51 - Selection of Sound output terminal (R/W)

```
Bit 7 - Output sound 4 to SO2 terminal
Bit 6 - Output sound 3 to SO2 terminal
Bit 5 - Output sound 2 to SO2 terminal
Bit 4 - Output sound 1 to SO2 terminal
Bit 3 - Output sound 4 to SO1 terminal
Bit 2 - Output sound 3 to SO1 terminal
Bit 1 - Output sound 2 to SO1 terminal
Bit 0 - Output sound 1 to SO1 terminal
```

FF26 - NR52 - Sound on/off

If your GB programs don't use sound then write 00h to this register to save 16% or more on GB power consumption. Disabeling the sound controller by clearing Bit 7 destroys the contents of all sound registers. Also, it is not possible to access any sound registers (execpt FF26) while the sound controller is disabled.

```
Bit 7 - All sound on/off  (0: stop all sound circuits) (Read/Write)
Bit 3 - Sound 4 ON flag (Read Only)
Bit 2 - Sound 3 ON flag (Read Only)
Bit 1 - Sound 2 ON flag (Read Only)
Bit 0 - Sound 1 ON flag (Read Only)
```

Bits 0-3 of this register are read only status bits, writing to these bits does NOT enable/disable sound. The flags get set when sound output is restarted by setting the Initial flag (Bit 7 in NR14-NR44), the flag remains set until the sound length has expired (if enabled). A volume envelopes which has decreased to zero volume will NOT cause the sound flag to go off.

Joypad Input

FF00 - P1/JOYP - Joypad (R/W)
The eight gameboy buttons/direction keys are arranged in form of a 2x4 matrix. Select either button or direction keys by writing to this register, then read-out bit 0-3.

```
Bit 7 - Not used
Bit 6 - Not used
Bit 5 - P15 Select Button Keys      (0=Select)
Bit 4 - P14 Select Direction Keys   (0=Select)
Bit 3 - P13 Input Down  or Start    (0=Pressed) (Read Only)
Bit 2 - P12 Input Up    or Select   (0=Pressed) (Read Only)
Bit 1 - P11 Input Left  or Button B (0=Pressed) (Read Only)
Bit 0 - P10 Input Right or Button A (0=Pressed) (Read Only)
```

Note: Most programs are repeatedly reading from this port several times (the first reads used as short delay, allowing the inputs to stabilize, and only the value from the last read actually used).

Usage in SGB software
Beside for normal joypad input, SGB games mis-use the joypad register to output SGB command packets to the SNES, also, SGB programs may read out gamepad states from up to four different joypads which can be connected to the SNES.
See SGB description for details.

INT 60 - Joypad Interrupt
Joypad interrupt is requested when any of the above Input lines changes from High to Low. Generally this should happen when a key becomes pressed (provided that the button/direction key is enabled by above Bit4/5), however, because of switch bounce, one or more High to Low transitions are usually produced both when pressing or releasing a key.

Using the Joypad Interrupt
It's more or less useless for programmers, even when selecting both buttons and direction keys simultaneously it still cannot recognize all keystrokes, because in that case a bit might be already held low by a button key, and pressing the corresponding direction key would thus cause no difference. The only meaningful purpose of the keystroke interrupt would be to terminate STOP (low power) standby state.
Also, the joypad interrupt does not appear to work with CGB and GBA hardware (the STOP function can be still terminated by joypad keystrokes though).

Serial Data Transfer (Link Cable)

FF01 - SB - Serial transfer data (R/W)
8 Bits of data to be read/written

FF02 - SC - Serial Transfer Control (R/W)

```
Bit 7 - Transfer Start Flag (0=No Transfer, 1=Start)
Bit 1 - Clock Speed (0=Normal, 1=Fast) ** CGB Mode Only **
Bit 0 - Shift Clock (0=External Clock, 1=Internal Clock)
```

The clock signal specifies the rate at which the eight data bits in SB (FF01) are transferred. When the gameboy is communicating with another gameboy (or other computer) then either one must supply internal clock, and the other one must use external clock.

Internal Clock

In Non-CGB Mode the gameboy supplies an internal clock of 8192Hz only (allowing to transfer about 1 KByte per second). In CGB Mode four internal clock rates are available, depending on Bit 1 of the SC register, and on whether the CGB Double Speed Mode is used:

```
  8192Hz -  1KB/s - Bit 1 cleared, Normal
 16384Hz -  2KB/s - Bit 1 cleared, Double Speed Mode
262144Hz - 32KB/s - Bit 1 set,     Normal
524288Hz - 64KB/s - Bit 1 set,     Double Speed Mode
```

External Clock

The external clock is typically supplied by another gameboy, but might be supplied by another computer (for example if connected to a PCs parallel port), in that case the external clock may have any speed. Even the old/monochrome gameboy is reported to recognizes external clocks of up to 500KHz. And there is no limitiation into the other direction - even when suppling an external clock speed of "1 bit per month", then the gameboy will still eagerly wait for the next bit(s) to be transferred. It isn't required that the clock pulses are sent at an regular interval either.

Timeouts

When using external clock then the transfer will not complete until the last bit is received. In case that the second gameboy isn't supplying a clock signal, if it gets turned off, or if there is no second gameboy connected at all) then transfer will never complete. For this reason the transfer procedure should use a timeout counter, and abort the communication if no response has been received during the timeout interval.

Delays and Synchronization

The gameboy that is using internal clock should always execute a small delay between each transfer, in order to ensure that the opponent gameboy has enough time to prepare itself for the next transfer, ie. the gameboy with external clock must have set its transfer start bit before the gameboy with internal clock starts the transfer. Alternately, the two gameboys could switch between internal and external clock for each transferred byte to ensure synchronization.

Transfer is initiated by setting the Transfer Start Flag. This bit is automatically set to 0 at the end of Transfer. Reading this bit can be used to determine if the transfer is still active.

INT 58 - Serial Interrupt

When the transfer has completed (ie. after sending/receiving 8 bits, if any) then an interrupt is requested by setting Bit 3 of the IF Register (FF0F). When that interrupt is enabled, then the Serial Interrupt vector at 0058 is called.

XXXXXX...

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

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.

During a transfer, a byte is shifted in at the same time that a byte is shifted out. The rate of the shift is determined by whether the clock source is internal or external.
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.

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
```

Timer and Divider Registers

FF04 - DIV - Divider Register (R/W)
This register is incremented at rate of 16384Hz (~16779Hz on SGB). In CGB Double Speed Mode it is incremented twice as fast, ie. at 32768Hz. Writing any value to this register resets it to 00h.

FF05 - TIMA - Timer counter (R/W)
This timer is incremented by a clock frequency specified by the TAC register ($FF07). When the value overflows (gets bigger than FFh) then it will be reset to the value specified in TMA (FF06), and an interrupt will be requested, as described below.

FF06 - TMA - Timer Modulo (R/W)
When the TIMA overflows, this data will be loaded.

FF07 - TAC - Timer Control (R/W)

```
  Bit 2    - Timer Stop  (0=Stop, 1=Start)
```

```
Bits 1-0 - Input Clock Select
           00:   4096 Hz    (~4194 Hz SGB)
           01: 262144 Hz  (~268400 Hz SGB)
           10:  65536 Hz   (~67110 Hz SGB)
           11:  16384 Hz   (~16780 Hz SGB)
```

INT 50 - Timer Interrupt
Each time when the timer overflows (ie. when TIMA gets bigger than FFh), then an interrupt is requested by setting Bit 2 in the IF Register (FF0F). When that interrupt is enabled, then the CPU will execute it by calling the timer interrupt vector at 0050h.

Note
The above described Timer is the built-in timer in the gameboy. It has nothing to do with the MBC3s battery buffered Real Time Clock - that's a completely different thing, described in the chapter about Memory Banking Controllers.

Interrupts

IME - Interrupt Master Enable Flag (Write Only)

```
0 - Disable all Interrupts
1 - Enable all Interrupts that are enabled in IE Register (FFFF)
```

The IME flag is used to disable all interrupts, overriding any enabled bits in the IE Register. It isn't possible to access the IME flag by using a I/O address, instead IME is accessed directly from the CPU, by the following opcodes/operations:

```
EI     ;Enable Interrupts  (ie. IME=1)
DI     ;Disable Interrupts (ie. IME=0)
RETI   ;Enable Ints & Return (same as the opcode combination EI, RET)
<INT>  ;Disable Ints & Call to Interrupt Vector
```

Whereas <INT> means the operation which is automatically executed by the CPU when it executes an interrupt.

FFFF - IE - Interrupt Enable (R/W)

```
Bit 0: V-Blank  Interrupt Enable  (INT 40h)  (1=Enable)
Bit 1: LCD STAT Interrupt Enable  (INT 48h)  (1=Enable)
Bit 2: Timer    Interrupt Enable  (INT 50h)  (1=Enable)
Bit 3: Serial   Interrupt Enable  (INT 58h)  (1=Enable)
Bit 4: Joypad   Interrupt Enable  (INT 60h)  (1=Enable)
```

FF0F - IF - Interrupt Flag (R/W)

```
Bit 0: V-Blank  Interrupt Request (INT 40h)  (1=Request)
Bit 1: LCD STAT Interrupt Request (INT 48h)  (1=Request)
Bit 2: Timer    Interrupt Request (INT 50h)  (1=Request)
Bit 3: Serial   Interrupt Request (INT 58h)  (1=Request)
Bit 4: Joypad   Interrupt Request (INT 60h)  (1=Request)
```

When an interrupt signal changes from low to high, then the corresponding bit in the IF register becomes set. For example, Bit 0 becomes set when the LCD controller enters into the V-Blank period.

Interrupt Requests
Any set bits in the IF register are only <requesting> an interrupt to be executed. The actual <execution> happens only if both the IME flag, and the corresponding bit in the IE register are set, otherwise the interrupt 'waits' until both IME and IE allow its execution.

Interrupt Execution
When an interrupt gets executed, the corresponding bit in the IF register becomes automatically reset by the CPU, and the IME flag becomes cleared (disabeling any further interrupts until the program re-enables the interrupts, typically by using the RETI instruction), and the corresponding Interrupt Vector (that are the addresses in range 0040h-0060h, as shown in IE and IF register decriptions above) becomes called.

Manually Requesting/Discarding Interrupts
As the CPU automatically sets and cleares the bits in the IF register it is usually not required to write to the IF register. However, the user may still do that in order to manually request (or discard) interrupts. As for real interrupts, a manually requested interrupt isn't executed unless/until IME and IE allow its execution.

Interrupt Priorities
In the following three situations it might happen that more than 1 bit in the IF register are set, requesting more than one interrupt at once:

```
1) More than one interrupt signal changed from Low
   to High at the same time.
2) Several interrupts have been requested during a
   time in which IME/IE didn't allow these interrupts
   to be executed directly.
3) The user has written a value with several "1" bits
   (for example 1Fh) to the IF register.
```

Provided that IME and IE allow the execution of more than one of the requested interrupts, then the interrupt with the highest priority becomes executed first. The priorities are ordered as the bits in the IE and IF registers, Bit 0 (V-Blank) having the highest priority, and Bit 4 (Joypad) having the lowest priority.

Nested Interrupts
The CPU automatically disables all other interrupts by setting IME=0 when it executes an interrupt. Usually IME remains zero until the interrupt procedure returns (and sets IME=1 by the RETI instruction). However, if you want any other interrupts of lower or higher (or same) priority to be allowed to be executed from inside of the interrupt procedure, then you can place an EI instruction into the interrupt procedure.

CGB Registers

Forward
This chapter describes only CGB (Color Gameboy) registers that didn't fit into normal categories - most CGB registers are described in the chapter about Video Display (Color Palettes, VRAM Bank, VRAM DMA Transfers, and changed meaning of Bit 0 of LCDC Control register). Also, a

changed bit is noted in the chapter about the Serial/Link port.

Unlocking CGB functions

When using any CGB registers (including those in the Video/Link chapters), you must first unlock CGB features by changing byte 0143h in the cartridge header. Typically use a value of 80h for games which support both CGB and monochrome gameboys, and C0h for games which work on CGBs only. Otherwise, the CGB will operate in monochrome "Non CGB" compatibility mode.

Detecting CGB (and GBA) functions

CGB hardware can be detected by examing the CPU accumulator (A-register) directly after startup. A value of 11h indicates CGB (or GBA) hardware, if so, CGB functions can be used (if unlocked, see above).
When A=11h, you may also examine Bit 0 of the CPUs B-Register to separate between CGB (bit cleared) and GBA (bit set), by that detection it is possible to use 'repaired' color palette data matching for GBA displays.

FF4D - KEY1 - CGB Mode Only - Prepare Speed Switch

```
Bit 7: Current Speed     (0=Normal, 1=Double) (Read Only)
Bit 0: Prepare Speed Switch (0=No, 1=Prepare) (Read/Write)
```

This register is used to prepare the gameboy to switch between CGB Double Speed Mode and Normal Speed Mode. The actual speed switch is performed by executing a STOP command after Bit 0 has been set. After that Bit 0 will be cleared automatically, and the gameboy will operate at the 'other' speed. The recommended speed switching procedure in pseudo code would be:

```
IF KEY1_BIT7 <> DESIRED_SPEED THEN
  IE=00H       ;(FFFF)=00h
  JOYP=30H     ;(FF00)=30h
  KEY1=01H     ;(FF4D)=01h
  STOP         ;STOP
ENDIF
```

The CGB is operating in Normal Speed Mode when it is turned on. Note that using the Double Speed Mode increases the power consumption, it would be recommended to use Single Speed whenever possible. However, the display will flicker (white) for a moment during speed switches, so this cannot be done permanentely.
In Double Speed Mode the following will operate twice as fast as normal:

```
The CPU (2.10 MHz, 1 Cycle = approx. 0.5us)
Timer and Divider Registers
Serial Port (Link Cable)
DMA Transfer to OAM
```

And the following will keep operating as usual:

```
LCD Video Controller
HDMA Transfer to VRAM
All Sound Timings and Frequencies
```

FF56 - RP - CGB Mode Only - Infrared Communications Port

This register allows to input and output data through the CGBs built-in Infrared Port. When reading data, bit 6 and 7 must be set (and obviously Bit 0 must be cleared - if you don't want to receive your own gameboys IR signal). After sending or receiving data you should reset the register to 00h to reduce battery power consumption again.

```
Bit 0:   Write Data   (0=LED Off, 1=LED On)             (Read/Write)
Bit 1:   Read Data    (0=Receiving IR Signal, 1=Normal) (Read Only)
Bit 6-7: Data Read Enable (0=Disable, 3=Enable)         (Read/Write)
```

Note that the receiver will adapt itself to the normal level of IR pollution in the air, so if you would send a LED ON signal for a longer period, then the receiver would treat that as normal (=OFF) after a while. For example, a Philips TV Remote Control sends a series of 32 LED ON/OFF pulses (length 10us ON, 17.5us OFF each) instead of a permanent 880us LED ON signal.
Even though being generally CGB compatible, the GBA does not include an infra-red port.

FF70 - SVBK - CGB Mode Only - WRAM Bank
In CGB Mode 32 KBytes internal RAM are available. This memory is divided into 8 banks of 4 KBytes each. Bank 0 is always available in memory at C000-CFFF, Bank 1-7 can be selected into the address space at D000-DFFF.

```
Bit 0-2  Select WRAM Bank (Read/Write)
```

Writing a value of 01h-07h will select Bank 1-7, writing a value of 00h will select Bank 1 either.

FF6C - Undocumented (FEh) - Bit 0 (Read/Write) - CGB Mode Only
FF72 - Undocumented (00h) - Bit 0-7 (Read/Write)
FF73 - Undocumented (00h) - Bit 0-7 (Read/Write)
FF74 - Undocumented (00h) - Bit 0-7 (Read/Write) - CGB Mode Only
FF75 - Undocumented (8Fh) - Bit 4-6 (Read/Write)
FF76 - Undocumented (00h) - Always 00h (Read Only)
FF77 - Undocumented (00h) - Always 00h (Read Only)
These are undocumented CGB Registers. The numbers in brackets () indicate the initial values. Purpose of these registers is unknown (if any).
Registers FF6C and FF74 are always FFh if the CGB is in Non CGB Mode.

SGB Functions

General Information

SGB Commands

SGB Description

General Description
Basically, the SGB (Super Gameboy) is an adapter cartridge that allows to play gameboy games on a SNES (Super Nintendo Entertainment System) gaming console. In detail, you plug the gameboy cartridge into the SGB cartridge, then plug the SGB cartridge into the SNES, and then connect the SNES to your TV Set. In result, games can be played and viewed on the TV Set, and are controlled by using the SNES joypad(s).

More Technical Description
The SGB cartridge just contains a normal gameboy CPU and normal gameboy video controller. Normally the video signal from this controller would be sent to the LCD screen, however, in this special case the SNES read out the video signal and displays it on the TV set by using a special SNES BIOS ROM which is located in the SGB cartridge. Also, normal gameboy sound output is forwared to the SNES and output to the TV Set, vice versa, joypad input is forwared from the SNES controller(s) to the gameboy joypad inputs.

Normal Monochrome Games
Any gameboy games which have been designed for normal monochrome handheld gameboys will work with the SGB hardware as well. The SGB will apply a four color palette to these games by replacing the normal four grayshades. The 160x144 pixel gamescreen is displayed in the middle of the 256x224 pixel SNES screen (the unused area is filled by a screen border bitmap). The user may access built-in menues, allowing to change color palette data, to select between several pre-defined borders, etc.

Games that have been designed to support SGB functions may also access the following additional features:

Colorized Game Screen
There's limited ability to colorize the gamescreen by assigning custom color palettes to each 20x18 display characters, however, this works mainly for static display data such like title screens or status bars, the 20x18 color attribute map is non-scrollable, and it is not possible to assign separate colors to moveable foreground sprites (OBJs), so that animated screen regions will be typically restricted to using a single palette of four colors only.

SNES Foreground Sprites
Up to 24 foreground sprites (OBJs) of 8x8 or 16x16 pixels, 16 colors can be displayed. When replacing (or just overlaying) the normal gameboy OBJs by SNES OBJs it'd be thus possible to display OBJs with other colors than normal background area. This method doesn't appear to be very popular, even though it appears to be quite easy to implement, however, the bottommost character line of the gamescreen will be masked out because this area is used to transfer OAM data to the SNES.

The SGB Border
The possibly most popular and most impressive feature is to replace the default SGB screen border by a custom bitmap which is stored in the game cartridge.

Multiple Joypads
Up to four joypads can be conected to the SNES, and SGB software may read-out each of these joypads separately, allowing up to four players to play the same game simultaneously. Unlike for multiplayer handheld games, this requires only one game cartridge and only one SGB/SNES, and no link cables are required, the downside is that all players must share the same display screen.

Sound Functions
Beside for normal gameboy sound, a number of digital sound effects is pre-defined in the SNES BIOS, these effects may be accessed quite easily. Programmers whom are familiar with SNES sounds may also access the SNES sound chip, or use the SNES MIDI engine directly in order to produce other sound effects or music.

Taking Control of the SNES CPU
Finally, it is possible to write program code or data into SNES memory, and to execute such program code by using the SNES CPU.

SGB System Clock
Because the SGB is synchronized to the SNES CPU, the gameboy system clock is directly chained to the SNES system clock. In result, the gameboy CPU, video controller, timers, and sound frequencies will be all operated approx 2.4% faster as by normal gameboys.
Basically, this should be no problem, and the game will just run a little bit faster. However sensitive musicians may notice that sound frequencies are a bit too high, programs that support SGB functions may avoid this effect by reducing frequencies of gameboy sounds when having detected SGB hardware.
Also, I think that I've heard that SNES models which use a 50Hz display refresh rate (rather than 60Hz) are resulting in respectively slower SGB/gameboy timings ???

SGB Unlocking and Detecting SGB Functions

Cartridge Header
SGB games are required to have a cartridge header with Nintendo and proper checksum just as normal gameboy games. Also, two special entries must be set in order to unlock SGB functions:

```
146h - SGB Flag - Must be set to 03h for SGB games
14Bh - Old Licensee Code - Must be set 33h for SGB games
```

When these entries aren't set, the game will still work just like all 'monochrome' gameboy games, but it cannot access any of the special SGB functions.

Detecting SGB hardware
The recommended detection method is to send a MLT_REQ command which enables two (or four) joypads. A normal handheld gameboy will ignore this command, a SGB will now return incrementing joypad IDs each time when deselecting keyboard lines (see MLT_REQ description for details). Now read-out joypad state/IDs several times, and if the ID-numbers are changing, then it is a SGB (a normal gameboy would typically always return 0Fh as ID). Finally, when not intending to use more than one joypad, send another MLT_REQ command in order to re-disable the multi-controller mode.
Detection works regardless of whether and how many joypads are physically connected to the SNES. However, detection works only when having

unlocked SGB functions in the cartridge header, as described above.

Separating between SGB and SGB2

It is also possible to separate between SGB and SGB2 models by examining the inital value of the accumulator (A-register) directly after startup.

```
01h  SGB or Normal Gameboy (DMG)
FFh  SGB2 or Pocket Gameboy
11h  CGB or GBA
```

Because values 01h and FFh are shared for both handhelds and SGBs, it is still required to use the above MLT_REQ detection procedure. As far as I know the SGB2 doesn't have any extra features which'd require separate SGB2 detection except for curiosity purposes, for example, the game "Tetris DX" chooses to display an alternate SGB border on SGB2s.

Reportedly, some SGB models include link ports (just like handheld gameboy) (my own SGB does not have such an port), possibly this feature is available in SGB2-type models only ???

SGB Command Packet Transfers

Command packets (aka Register Files) are transferred from the gameboy to the SNES by using P14 and P15 output lines of the JOYPAD register (FF00h), these lines are normally used to select the two rows in the gameboy keyboard matrix (which still works).

Transferring Bits

A command packet transfer must be initiated by setting both P14 and P15 to LOW, this will reset and start the SNES packet receiving program. Data is then transferred (LSB first), setting P14=LOW will indicate a "0" bit, and setting P15=LOW will indicate a "1" bit. For example:

```
      RESET 0   0   1   1   0   1   0
P14  --_---_---_-----------_-------_--...
P15  --_-----------_---_-------_------...
```

Data and reset pulses must be kept LOW for at least 5us. P14 and P15 must be kept both HIGH for at least 15us between any pulses.
Obviously, it'd be no good idea to access the JOYPAD register during the transfer, for example, in case that your VBlank interrupt procedure reads-out joypad states each frame, be sure to disable that interrupt during the transfer (or disable only the joypad procedure by using a software flag).

Transferring Packets

Each packet is invoked by a RESET pulse, then 128 bits of data are transferred (16 bytes, LSB of first byte first), and finally, a "0"-bit must be transferred as stop bit. The structure of normal packets is:

```
 1 PULSE Reset
 1 BYTE  Command Code*8+Length
15 BYTES Parameter Data
 1 BIT   Stop Bit (0)
```

The above 'Length' indicates the total number of packets (1-7, including the first packet) which will be sent, ie. if more than 15 parameter bytes are used, then further packet(s) will follow, as such:

```
 1 PULSE Reset
16 BYTES Parameter Data
```

```
1 BIT   Stop Bit (0)
```

By using all 7 packets, up to 111 data bytes (15+16*6) may be sent.
Unused bytes at the end of the last packet must be set to zero.
A 60ms (4 frames) delay should be invoked between each packet transfer.

SGB VRAM Transfers

Overview
Beside for the packet transfer method, larger data blocks of 4KBytes can be transferred by using the video signal. These transfers are invoked by first sending one of the commands with the ending _TRN (by using normal packet transfer), the 4K data block is then read-out by the SNES from gameboy display memory during the next frame.

Transfer Data
Normally, transfer data should be stored at 8000h-8FFFh in gameboy VRAM,
even though the SNES receives the data in from display scanlines, it will automatically re-produce the same ordering of bits and bytes, as being originally stored at 8000h-8FFFh in gameboy memory.

Preparing the Display
The above method works only when recursing the following things: BG Map must display unsigned characters 00h-FFh on the screen; 00h..13h in first line, 14h..27h in next line, etc. The gameboy display must be enabled, the display may not be scrolled, OBJ sprites should not overlap the background tiles, the BGP palette register must be set to E4h.

Transfer Time
Note that the transfer data should be prepared in VRAM <before> sending the transfer command packet. The actual transfer starts at the beginning of the next frame after the command has been sent, and the transfer ends at the end of the 5th frame after the command has been sent (not counting the frame in which the command has been sent).

Avoiding Screen Garbage
The display will contain 'garbage' during the transfer, this dirt-effect can be avoided by freezing the screen (in the state which has been displayed before the transfer) by using the MASK_EN command.
Of course, this works only when actually executing the game on a SGB (and not on normal handheld gameboys), it'd be thus required to detect the presence of SGB hardware before blindly sending VRAM data.

SGB Command Summary

SGB System Command Table

```
Code Name      Expl.
```

```
00   PAL01     Set SGB Palette 0,1 Data
01   PAL23     Set SGB Palette 2,3 Data
02   PAL03     Set SGB Palette 0,3 Data
03   PAL12     Set SGB Palette 1,2 Data
04   ATTR_BLK  "Block" Area Designation Mode
05   ATTR_LIN  "Line" Area Designation Mode
06   ATTR_DIV  "Divide" Area Designation Mode
07   ATTR_CHR  "1CHR" Area Designation Mode
08   SOUND     Sound On/Off
09   SOU_TRN   Transfer Sound PRG/DATA
0A   PAL_SET   Set SGB Palette Indirect
0B   PAL_TRN   Set System Color Palette Data
0C   ATRC_EN   Enable/disable Attraction Mode
0D   TEST_EN   Speed Function
0E   ICON_EN   SGB Function
0F   DATA_SND  SUPER NES WRAM Transfer 1
10   DATA_TRN  SUPER NES WRAM Transfer 2
11   MLT_REG   Controller 2 Request
12   JUMP      Set SNES Program Counter
13   CHR_TRN   Transfer Character Font Data
14   PCT_TRN   Set Screen Data Color Data
15   ATTR_TRN  Set Attribute from ATF
16   ATTR_SET  Set Data to ATF
17   MASK_EN   Game Boy Window Mask
18   OBJ_TRN   Super NES OBJ Mode
```

SGB Color Palettes Overview

Available SNES Palettes
The SGB/SNES provides 8 palettes of 16 colors each, each color may be defined out of a selection of 34768 colors (15 bit). Palettes 0-3 are used to colorize the gamescreen, only the first four colors of each of these palettes are used. Palettes 4-7 are used for the SGB Border, all 16 colors of each of these palettes may be used.

Color 0 Restriction
Color 0 of each of the eight palettes is transparent, causing the backdrop color to be displayed instead. The backdrop color is typically defined by the most recently color being assigned to Color 0 (regardless of the palette number being used for that operation).
Effectively, gamescreen palettes can have only three custom colors each, and SGB border palettes only 15 colors each, additionally, color 0 can be used for for all palettes, which will then all share the same color though.

Translation of Grayshades into Colors
Because the SGB/SNES reads out the gameboy video controllers display signal, it translates the different grayshades from the signal into SNES colors as such:

```
White       -->  Color 0
```

```
Light Gray  -->  Color 1
Dark Gray   -->  Color 2
Black       -->  Color 3
```

Note that gameboy colors 0-3 are assigned to user-selectable grayshades by the gameboys BGP, OBP1, and OBP2 registers. There is thus no fixed relationship between gameboy colors 0-3 and SNES colors 0-3.

Using Gameboy BGP/OBP Registers
A direct translation of color 0-3 into color 0-3 may be produced by setting BGP/OBP registers to a value of 0E4h each. However, in case that your program uses black background for example, then you may internally assign background as "White" at the gameboy side by BGP/OBP registers (which is then interpreted as SNES color 0, which is shared for all SNES palettes). The advantage is that you may define Color 0 as Black at the SNES side, and may assign custom colors for Colors 1-3 of each SNES palette.

System Color Palette Memory
Beside for the actually visible palettes, up to 512 palettes of 4 colors each may be defined in SNES RAM. Basically, this is completely irrelevant because the palettes are just stored in RAM whithout any relationship to the displayed picture, anyways, these pre-defined colors may be transferred to actually visible palettes slightly faster as when transferring palette data by separate command packets.

SGB Palette Commands

SGB Command 00h - PAL01
Transmit color data for SGB palette 0, color 0-3, and for SGB palette 1, color 1-3 (without separate color 0).

```
Byte  Content
0     Command*8+Length (fixed length=01h)
1-E   Color Data for 7 colors of 2 bytes (16bit) each:
        Bit 0-4   - Red Intensity   (0-31)
        Bit 5-9   - Green Intensity (0-31)
        Bit 10-14 - Blue Intensity  (0-31)
        Bit 15    - Not used (zero)
F     Not used (00h)
```

The value transferred as color 0 will be applied for all eight palettes.

SGB Command 01h - PAL23
Same as above PAL01, but for Palettes 2 and 3 respectively.

SGB Command 02h - PAL03
Same as above PAL01, but for Palettes 0 and 3 respectively.

SGB Command 03h - PAL12
Same as above PAL01, but for Palettes 1 and 2 respectively.

SGB Command 0Ah - PAL_SET

Used to copy pre-defined palette data from SGB system color palette to actual SGB palette.

```
Byte  Content
0     Command*8+Length (fixed length=1)
1-2   System Palette number for SGB Color Palette 0 (0-511)
3-4   System Palette number for SGB Color Palette 1 (0-511)
5-6   System Palette number for SGB Color Palette 2 (0-511)
7-8   System Palette number for SGB Color Palette 3 (0-511)
9     Attribute File
        Bit 0-5 - Attribute File Number (00h-2Ch) (Used only if Bit7=1)
        Bit 6   - Cancel Mask           (0=No change, 1=Yes)
        Bit 7   - Use Attribute File    (0=No, 1=Apply above ATF Number)
A-F   Not used (zero)
```

Before using this function, System Palette data should be initialized by PAL_TRN command, and (when used) Attribute File data should be initialized by ATTR_TRN.

SGB Command 0Bh - PAL_TRN

Used to initialize SGB system color palettes in SNES RAM.

System color palette memory contains 512 pre-defined palettes, these palettes do not directly affect the display, however, the PAL_SET command may be later used to transfer four of these 'logical' palettes to actual visible 'physical' SGB palettes. Also, the OBJ_TRN function will use groups of 4 System Color Palettes (4*4 colors) for SNES OBJ palettes (16 colors).

```
Byte  Content
0     Command*8+Length (fixed length=1)
1-F   Not used (zero)
```

The palette data is sent by VRAM-Transfer (4 KBytes).

```
000-FFF  Data for System Color Palette 0-511
```

Each Palette consists of four 16bit-color definitions (8 bytes).

Note: The data is stored at 3000h-3FFFh in SNES memory.

SGB Color Attribute Commands

SGB Command 04h - ATTR_BLK

Used to specify color attributes for the inside or outside of one or more rectangular screen regions.

```
Byte  Content
0     Command*8+Length (length=1..7)
1     Number of Data Sets (01h..12h)
2-7   Data Set #1
        Byte 0 - Control Code (0-7)
          Bit 0 - Change Colors inside of surrounded area     (1=Yes)
          Bit 1 - Change Colors of surrounding character line (1=Yes)
          Bit 2 - Change Colors outside of surrounded area    (1=Yes)
          Bit 3-7 - Not used (zero)
```

```
            Exception: When changing only the Inside or Outside, then the
            Surrounding line becomes automatically changed to same color.
          Byte 1 - Color Palette Designation
            Bit 0-1 - Palette Number for inside of surrounded area
            Bit 2-3 - Palette Number for surrounding character line
            Bit 4-5 - Palette Number for outside of surrounded area
            Bit 6-7 - Not used (zero)
          Data Set Byte 2 - Coordinate X1 (left)
          Data Set Byte 3 - Coordinate Y1 (upper)
          Data Set Byte 4 - Coordinate X2 (right)
          Data Set Byte 5 - Coordinate Y2 (lower)
            Specifies the coordinates of the surrounding rectangle.
  8-D   Data Set #2 (if any)
  E-F   Data Set #3 (continued at 0-3 in next packet) (if any)
```

When sending three or more data sets, data is continued in further packet(s). Unused bytes at the end of the last packet should be set to zero. The format of the separate Data Sets is described below.

SGB Command 05h - ATTR_LIN

Used to specify color attributes of one or more horizontal or vertical character lines.

```
  Byte  Content
  0     Command*8+Length (length=1..7)
  1     Number of Data Sets (01h..6Eh) (one byte each)
  2     Data Set #1
          Bit 0-4 - Line Number    (X- or Y-coordinate, depending on bit 7)
          Bit 5-6 - Palette Number (0-3)
          Bit 7   - H/V Mode Bit   (0=Vertical line, 1=Horizontal Line)
  3     Data Set #2 (if any)
  4     Data Set #3 (if any)
  etc.
```

When sending 15 or more data sets, data is continued in further packet(s). Unused bytes at the end of the last packet should be set to zero. The format of the separate Data Sets (one byte each) is described below.
The length of each line reaches from one end of the screen to the other end. In case that some lines overlap each other, then lines from lastmost data sets will overwrite lines from previous data sets.

SGB Command 06h - ATTR_DIV

Used to split the screen into two halfes, and to assign separate color attributes to each half, and to the division line between them.

```
  Byte  Content
  0     Command*8+Length   (fixed length=1)
  1     Color Palette Numbers and H/V Mode Bit
          Bit 0-1  Palette Number below/right of division line
          Bit 2-3  Palette Number above/left of division line
          Bit 4-5  Palette Number for division line
          Bit 6    H/V Mode Bit  (0=split left/right, 1=split above/below)
  2     X- or Y-Coordinate (depending on H/V bit)
  3-F   Not used (zero)
```

SGB Command 07h - ATTR_CHR
Used to specify color attributes for separate characters.

```
Byte  Content
0     Command*8+Length (length=1..6)
1     Beginning X-Coordinate
2     Beginning Y-Coordinate
3-4   Number of Data Sets (1-360)
5     Writing Style   (0=Left to Right, 1=Top to Bottom)
6     Data Sets 1-4   (Set 1 in MSBs, Set 4 in LSBs)
7     Data Sets 5-8   (if any)
8     Data Sets 9-12  (if any)
etc.
```

When sending 41 or more data sets, data is continued in further packet(s). Unused bytes at the end of the last packet should be set to zero. Each data set consists of two bits, indicating the palette number for one character.
Depending on the writing style, data sets are written from left to right, or from top to bottom. In either case the function wraps to the next row/column when reaching the end of the screen.

SGB Command 15h - ATTR_TRN
Used to initialize Attribute Files (ATFs) in SNES RAM. Each ATF consists of 20x18 color attributes for the gameboy screen. This function does not directly affect display attributes. Instead, one of the defined ATFs may be copied to actual display memory at a later time by using ATTR_SET or PAL_SET functions.

```
Byte  Content
0     Command*8+Length (fixed length=1)
1-F   Not used (zero)
```

The ATF data is sent by VRAM-Transfer (4 KBytes).

```
000-FD1  Data for ATF0 through ATF44 (4050 bytes)
FD2-FFF  Not used
```

Each ATF consists of 90 bytes, that are 5 bytes (20x2bits) for each of the 18 character lines of the gameboy window. The two most significant bits of the first byte define the color attribute (0-3) for the first character of the first line, the next two bits the next character, and so on.

SGB Command 16h - ATTR_SET
Used to transfer attributes from Attribute File (ATF) to gameboy window.

```
Byte  Content
0     Command*8+Length (fixed length=1)
1     Attribute File Number (00-2Ch), Bit 6=Cancel Mask
2-F   Not used (zero)
```

When above Bit 6 is set, the gameboy screen becomes re-enabled after the transfer (in case it has been disabled/frozen by MASK_EN command).
Note: The same functions may be (optionally) also included in PAL_SET commands, as described in the chapter about Color Palette Commands.

SGB Sound Functions

SGB Command 08h - SOUND

Used to start/stop internal sound effect, start/stop sound using internal tone data.

```
Byte  Content
0     Command*8+Length (fixed length=1)
1     Sound Effect A (Port 1) Decrescendo 8bit Sound Code
2     Sound Effect B (Port 2) Sustain     8bit Sound Code
3     Sound Effect Attributes
        Bit 0-1 - Sound Effect A Pitch  (0..3=Low..High)
        Bit 2-3 - Sound Effect A Volume (0..2=High..Low, 3=Mute on)
        Bit 4-5 - Sound Effect B Pitch  (0..3=Low..High)
        Bit 6-7 - Sound Effect B Volume (0..2=High..Low, 3=Not used)
4     Music Score Code (must be zero if not used)
5-F   Not used (zero)
```

See Sound Effect Tables below for a list of available pre-defined effects.

"Notes"

1) Mute is only active when both bits D2 and D3 are 1.

2) When the volume is set for either Sound Effect A or Sound Effect B, mute is turned off.

3) When Mute on/off has been executed, the sound fades out/fades in.

4) Mute on/off operates on the (BGM) which is reproduced by Sound Effect A, Sound Effect B, and the Super NES APU. A "mute off" flag does not exist by itself. When mute flag is set, volume and pitch of Sound Effect A (port 1) and Sound Effect B (port 2) must be set.

SGB Command 09h - SOU_TRN

Used to transfer sound code or data to SNES Audio Processing Unit memory (APU-RAM).

```
Byte  Content
0     Command*8+Length (fixed length=1)
1-F   Not used (zero)
```

The sound code/data is sent by VRAM-Transfer (4 KBytes).

```
000      One (or two ???) 16bit expression(s ???) indicating the
         transfer destination address and transfer length.
...-...  Transfer Data
...-FFF  Remaining bytes not used
```

Possible destinations in APU-RAM are:

```
0400h-2AFFh  APU-RAM Program Area (9.75KBytes)
2B00h-4AFFh  APU-RAM Sound Score Area (8Kbytes)
4DB0h-EEFFh  APU-RAM Sampling Data Area (40.25 Kbytes)
```

This function may be used to take control of the SNES sound chip, and/or to access the SNES MIDI engine. In either case it requires deeper knowledge of SNES sound programming.

SGB Sound Effect A/B Tables

Below lists the digital sound effects that are pre-defined in the SGB/SNES BIOS, and which can be used with the SGB "SOUND" Command. Effect A and B may be simultaneously reproduced.

The P-column indicates the recommended Pitch value, the V-column indicates the numbers of Voices used. Sound Effect A uses voices 6,7. Sound

Effect B uses voices 0,1,4,5. Effects that use less voices will use only the upper voices (eg. 4,5 for Effect B with only two voices).

Sound Effect A Flag Table

Code	Description	P	V	Code	Description	P	V
00	Dummy flag, re-trigger	-	2	18	Fast Jump	3	1
80	Effect A, stop/silent	-	2	19	Jet (rocket) takeoff	0	1
01	Nintendo	3	1	1A	Jet (rocket) landing	0	1
02	Game Over	3	2	1B	Cup breaking	2	2
03	Drop	3	1	1C	Glass breaking	1	2
04	OK ... A	3	2	1D	Level UP	2	2
05	OK ... B	3	2	1E	Insert air	1	1
06	Select...A	3	2	1F	Sword swing	1	1
07	Select...B	3	1	20	Water falling	2	1
08	Select...C	2	2	21	Fire	1	1
09	Mistake...Buzzer	2	1	22	Wall collapsing	1	2
0A	Catch Item	2	2	23	Cancel	1	2
0B	Gate squeaks 1 time	2	2	24	Walking	1	2
0C	Explosion...small	1	2	25	Blocking strike	1	2
0D	Explosion...medium	1	2	26	Picture floats on & off	3	2
0E	Explosion...large	1	2	27	Fade in	0	2
0F	Attacked...A	3	1	28	Fade out	0	2
10	Attacked...B	3	2	29	Window being opened	1	2
11	Hit (punch)...A	0	2	2A	Window being closed	0	2
12	Hit (punch)...B	0	2	2B	Big Laser	3	2
13	Breath in air	3	2	2C	Stone gate closes/opens	0	2
14	Rocket Projectile...A	3	2	2D	Teleportation	3	1
15	Rocket Projectile...B	3	2	2E	Lightning	0	2
16	Escaping Bubble	2	1	2F	Earthquake	0	2
17	Jump	3	1	30	Small Laser	2	2

Sound effect A is used for formanto sounds (percussion sounds).

Sound Effect B Flag Table

Code	Description	P	V	Code	Description	P	V
00	Dummy flag, re-trigger	-	4	0D	Waterfall	2	2
80	Effect B, stop/silent	-	4	0E	Small character running	3	1
01	Applause...small group	2	1	0F	Horse running	3	1
02	Applause...medium group	2	2	10	Warning sound	1	1
03	Applause...large group	2	4	11	Approaching car	0	1
04	Wind	1	2	12	Jet flying	1	1
05	Rain	1	1	13	UFO flying	2	1
06	Storm	1	3	14	Electromagnetic waves	0	1
07	Storm with wind/thunder	2	4	15	Score UP	3	1
08	Lightning	0	2	16	Fire	2	1
09	Earthquake	0	2	17	Camera shutter, formanto	3	4
0A	Avalanche	0	2	18	Write, formanto	0	1
0B	Wave	0	1	19	Show up title, formanto	0	1
0C	River	3	2				

Sound effect B is mainly used for looping sounds (sustained sounds).

SGB System Control Commands

SGB Command 17h - MASK_EN

Used to mask the gameboy window, among others this can be used to freeze the gameboy screen before transferring data through VRAM (the SNES then keeps displaying the gameboy screen, even though VRAM doesn't contain meaningful display information during the transfer).

```
Byte  Content
0     Command*8+Length (fixed length=1)
1     Gameboy Screen Mask (0-3)
        0  Cancel Mask   (Display activated)
        1  Freeze Screen (Keep displaying current picture)
        2  Blank Screen  (Black)
        3  Blank Screen  (Color 0)
2-F   Not used (zero)
```

Freezing works only if the SNES has stored a picture, ie. if necessary wait one or two frames before freezing (rather than freezing directly after having displayed the picture).
The Cancel Mask function may be also invoked (optionally) by completion of PAL_SET and ATTR_SET commands.

SGB Command 0Ch - ATRC_EN

Used to enable/disable Attraction mode. It is totally unclear what an attraction mode is ???, but it is enabled by default.

```
Byte  Content
0     Command*8+Length    (fixed length=1)
1     Attraction Disable  (0=Enable, 1=Disable)
2-F   Not used (zero)
```

SGB Command 0Dh - TEST_EN

Used to enable/disable test mode for "SGB-CPU variable clock speed function". This function is disabled by default.

```
Byte  Content
0     Command*8+Length    (fixed length=1)
1     Test Mode Enable    (0=Disable, 1=Enable)
2-F   Not used (zero)
```

Maybe intended to determine whether SNES operates at 50Hz or 60Hz display refresh rate ??? Possibly result can be read-out from joypad register ???

SGB Command 0Eh - ICON_EN

Used to enable/disable ICON function. Possibly meant to enable/disable SGB/SNES popup menues which might otherwise activated during gameboy game play. By default all functions are enabled (0).

```
Byte  Content
0     Command*8+Length    (fixed length=1)
1     Disable Bits
```

```
          Bit 0 - Use of SGB-Built-in Color Palettes    (1=Disable)
          Bit 1 - Controller Set-up Screen    (0=Enable, 1=Disable)
          Bit 2 - SGB Register File Transfer (0=Receive, 1=Disable)
          Bit 3-6 - Not used (zero)
  2-F   Not used (zero)
```

Above Bit 2 will suppress all further packets/commands when set, this might be useful when starting a monochrome game from inside of the SGB-menu of a multi-gamepak which contains a collection of different games.

SGB Command 0Fh - DATA_SND

Used to write one or more bytes directly into SNES Work RAM.

```
  Byte  Content
  0     Command*8+Length    (fixed length=1)
  1     SNES Destination Address, low
  2     SNES Destination Address, high
  3     SNES Destination Address, bank number
  4     Number of bytes to write (01h-0Bh)
  5     Data Byte #1
  6     Data Byte #2 (if any)
  7     Data Byte #3 (if any)
  etc.
```

Unused bytes at the end of the packet should be set to zero, this function is restricted to a single packet, so that not more than 11 bytes can be defined at once.
Free Addresses in SNES memory are Bank 0 1800h-1FFFh, Bank 7Fh 0000h-FFFFh.

SGB Command 10h - DATA_TRN

Used to transfer binary code or data directly into SNES RAM.

```
  Byte  Content
  0     Command*8+Length    (fixed length=1)
  1     SNES Destination Address, low
  2     SNES Destination Address, high
  3     SNES Destination Address, bank number
  4-F   Not used (zero)
```

The data is sent by VRAM-Transfer (4 KBytes).

```
  000-FFF  Data
```

Free Addresses in SNES memory are Bank 0 1800h-1FFFh, Bank 7Fh 0000h-FFFFh. The transfer length is fixed at 4KBytes ???, so that directly writing to the free 2KBytes at 0:1800h would be a not so good idea ???

SGB Command 12h - JUMP

Used to set the SNES program counter to a specified address. Optionally, it may be used to set a new address for the SNES NMI handler, the NMI handler remains unchanged if all bytes 4-6 are zero.

```
  Byte  Content
  0     Command*8+Length    (fixed length=1)
  1     SNES Program Counter, low
  2     SNES Program Counter, high
```

```
3     SNES Program Counter, bank number
4     SNES NMI Handler, low
5     SNES NMI Handler, high
6     SNES NMI Handler, bank number
7-F   Not used, zero
```

Note: The game "Space Invaders 94" uses this function when selecting "Arcade mode" to execute SNES program code which has been previously transferred from the SGB to the SNES. The type of the CPU which is used in the SNES is unknown ???

SGB Multiplayer Command

SGB Command 11h - MLT_REQ

Used to request multiplayer mode (ie. input from more than one joypad).
Because this function provides feedback from the SGB/SNES to the gameboy program, it is also used to detect SGB hardware.

```
Byte  Content
0     Command*8+Length    (fixed length=1)
1     Multiplayer Control (0-3) (Bit0=Enable, Bit1=Two/Four Players)
        0 = One player
        1 = Two players
        3 = Four players
2-F   Not used (zero)
```

In one player mode, the second joypad (if any) is used for the SGB system program. In two player mode, both joypads are used for the game. Because SNES have only two joypad sockets, four player mode requires an external "Multiplayer 5" adapter.

Reading Multiple Controllers (Joypads)

When having enabled multiple controllers by MLT_REQ, data for each joypad can be read out through JOYPAD register (FF00) as follows: First set P14 and P15 both HIGH (deselect both Buttons and Cursor keys), you can now read the lower 4bits of FF00 which indicate the joypad ID for the following joypad input:

```
0Fh  Joypad 1
0Eh  Joypad 2
0Dh  Joypad 3
0Ch  Joypad 4
```

Next, set P14 and P15 low (one after each other) to select Buttons and Cursor lines, and read-out joypad state as normally. When completed, set P14 and P15 back HIGH, this automatically increments the joypad number (or restarts counting once reached the lastmost joypad). Repeat the procedure until you have read-out states for all two (or four) joypads.

SGB Border and OBJ Commands

SGB Command 13h - CHR_TRN

Used to transfer tile data (characters) to SNES Tile memory in VRAM. This normally used to define BG tiles for the SGB Border (see PCT_TRN), but might be also used to define moveable SNES foreground sprites (see OBJ_TRN).

```
Byte  Content
0     Command*8+Length    (fixed length=1)
1     Tile Transfer Destination
        Bit 0   - Tile Numbers   (0=Tiles 00h-7Fh, 1=Tiles 80h-FFh)
        Bit 1   - Tile Type      (0=BG Tiles, 1=OBJ Tiles)
        Bit 2-7 - Not used (zero)
2-F   Not used (zero)
```

The tile data is sent by VRAM-Transfer (4 KBytes).

```
000-FFF  Bitmap data for 128 Tiles
```

Each tile occupies 16bytes (8x8 pixels, 16 colors each).
When intending to transfer more than 128 tiles, call this function twice (once for tiles 00h-7Fh, and once for tiles 80h-FFh). Note: The BG/OBJ Bit seems to have no effect and writes to the same VRAM addresses for both BG and OBJ ???

SGB Command 14h - PCT_TRN

Used to transfer tile map data and palette data to SNES BG Map memory in VRAM to be used for the SGB border. The actual tiles must be separately transferred by using the CHR_TRN function.

```
Byte  Content
0     Command*8+Length    (fixed length=1)
1-F   Not used (zero)
```

The map data is sent by VRAM-Transfer (4 KBytes).

```
000-7FF  BG Map 32x32 Entries of 16bit each (2048 bytes)
800-87F  BG Palette Data (Palettes 4-7, each 16 colors of 16bits each)
880-FFF  Not used, don't care
```

Each BG Map Entry consists of a 16bit value as such:

```
Bit 0-9   - Character Number (use only 00h-FFh, upper 2 bits zero)
Bit 10-12 - Palette Number   (use only 4-7, officially use only 4-6)
Bit 13    - BG Priority      (use only 0)
Bit 14    - X-Flip           (0=Normal, 1=Mirror horizontally)
Bit 15    - Y-Flip           (0=Normal, 1=Mirror vertically)
```

Even though 32x32 map entries are transferred, only upper 32x28 are actually used (256x224 pixels, SNES screen size). The 20x18 entries in the center of the 32x28 area should be set to 0000h as transparent space for the gameboy window to be displayed inside. Reportedly, non-transparent border data will cover the gameboy window.

SGB Command 18h - OBJ_TRN

Used to transfer OBJ attributes to SNES OAM memory. Unlike all other functions with the ending _TRN, this function does not use the usual one-shot 4KBytes VRAM transfer method.
Instead, when enabled (below execute bit set), data is permanently (each frame) read out from the lower character line of the gameboy screen. To suppress garbage on the display, the lower line is masked, and only the upper 20x17 characters of the gameboy window are used - the masking method is unknwon - frozen, black, or recommended to be covered by the SGB border, or else ??? Also, when the function is enabled, "system attract mode is not performed" - whatever that means ???

```
Byte  Content
0     Command*8+Length (fixed length=1)
1     Control Bits
        Bit 0   - SNES OBJ Mode enable (0=Cancel, 1=Enable)
        Bit 1   - Change OBJ Color     (0=No, 1=Use definitions below)
        Bit 2-7 - Not used (zero)
2-3   System Color Palette Number for OBJ Palette 4 (0-511)
4-5   System Color Palette Number for OBJ Palette 5 (0-511)
6-7   System Color Palette Number for OBJ Palette 6 (0-511)
8-9   System Color Palette Number for OBJ Palette 7 (0-511)
        These color entries are ignored if above Control Bit 1 is zero.
        Because each OBJ palette consists of 16 colors, four system
        palette entries (of 4 colors each) are transferred into each
        OBJ palette. The system palette numbers are not required to be
        aligned to a multiple of four, and will wrap to palette number
        0 when exceeding 511. For example, a value of 511 would copy
        system palettes 511, 0, 1, 2 to the SNES OBJ palette.
A-F   Not used (zero)
```

The recommended method is to "display" gameboy BG tiles F9h..FFh from left to right as first 7 characters of the bottom-most character line of the gameboy screen. As for normal 4KByte VRAM transfers, this area should not be scrolled, should not be overlapped by gameboy OBJs, and the gameboy BGP palette register should be set up properly. By following that method, SNES OAM data can be defined in the 70h bytes of the gameboy BG tile memory at following addresses:

```
8F90-8FEF  SNES OAM, 24 Entries of 4 bytes each (96 bytes)
8FF0-8FF5  SNES OAM MSBs, 24 Entries of 2 bits each (6 bytes)
8FF6-8FFF  Not used, don't care (10 bytes)
```

The format of SNES OAM Entries is:

```
Byte 0  OBJ X-Position (0-511, MSB is separately stored, see below)
Byte 1  OBJ Y-Position (0-255)
Byte 2-3  Attributes (16bit)
  Bit 0-8    Tile Number     (use only 00h-FFh, upper bit zero)
  Bit 9-11   Palette Number  (use only 4-7)
  Bit 12-13  OBJ Priority    (use only 3)
  Bit 14     X-Flip          (0=Normal, 1=Mirror horizontally)
  Bit 15     Y-Flip          (0=Normal, 1=Mirror vertically)
```

The format of SNES OAM MSB Entries is:

```
Actually, the format is unknown ??? However, 2 bits are used per entry:
One bit is the most significant bit of the OBJ X-Position.
The other bit specifies the OBJ size (8x8 or 16x16 pixels).
```

CPU Registers and Flags

Registers

```
16bit Hi   Lo   Name/Function
```

```
AF    A    -    Accumulator & Flags
BC    B    C    BC
DE    D    E    DE
HL    H    L    HL
SP    -    -    Stack Pointer
PC    -    -    Program Counter/Pointer
```

As shown above, most registers can be accessed either as one 16bit register, or as two separate 8bit registers.

The Flag Register (lower 8bit of AF register)

```
Bit  Name  Set Clr  Expl.
7    zf    Z   NZ   Zero Flag
6    n     -   -    Add/Sub-Flag (BCD)
5    h     -   -    Half Carry Flag (BCD)
4    cy    C   NC   Carry Flag
3-0  -     -   -    Not used (always zero)
```

Conatins the result from the recent instruction which has affected flags.

The Zero Flag (Z)
This bit becomes set (1) if the result of an operation has been zero (0). Used for conditional jumps.

The Carry Flag (C, or Cy)
Becomes set when the result of an addition became bigger than FFh (8bit) or FFFFh (16bit). Or when the result of a subtraction or comparision became less than zero (much as for Z80 and 80x86 CPUs, but unlike as for 65XX and ARM CPUs). Also the flag becomes set when a rotate/shift operation has shifted-out a "1"-bit.
Used for conditional jumps, and for instructions such like ADC, SBC, RL, RLA, etc.

The BCD Flags (N, H)
These flags are (rarely) used for the DAA instruction only, N Indicates whether the previous instruction has been an addition or subtraction, and H indicates carry for lower 4bits of the result, also for DAA, the C flag must indicate carry for upper 8bits.
After adding/subtracting two BCD numbers, DAA is intended to convert the result into BCD format; BCD numbers are ranged from 00h to 99h rather than 00h to FFh.
Because C and H flags must contain carry-outs for each digit, DAA cannot be used for 16bit operations (which have 4 digits), or for INC/DEC operations (which do not affect C-flag).

CPU Instruction Set

Tables below specify the mnemonic, opcode bytes, clock cycles, affected flags (ordered as znhc), and explanatation.
The timings assume a CPU clock frequency of 4.194304 MHz (or 8.4
MHz for CGB in double speed mode), as all gameboy timings are divideable
by 4, many people specify timings and clock frequency divided by 4.

GMB 8bit-Loadcommands

```
ld   r,r         xx         4 ----  r=r
ld   r,n         xx nn      8 ----  r=n
ld   r,(HL)      xx         8 ----  r=(HL)
ld   (HL),r      7x         8 ----  (HL)=r
ld   (HL),n      36 nn     12 ----
ld   A,(BC)      0A         8 ----
ld   A,(DE)      1A         8 ----
ld   A,(nn)      FA        16 ----
ld   (BC),A      02         8 ----
ld   (DE),A      12         8 ----
ld   (nn),A      EA        16 ----
ld   A,(FF00+n)  F0 nn     12 ----  read from io-port n (memory FF00+n)
ld   (FF00+n),A  E0 nn     12 ----  write to io-port n (memory FF00+n)
ld   A,(FF00+C)  F2         8 ----  read from io-port C (memory FF00+C)
ld   (FF00+C),A  E2         8 ----  write to io-port C (memory FF00+C)
ldi  (HL),A      22         8 ----  (HL)=A, HL=HL+1
ldi  A,(HL)      2A         8 ----  A=(HL), HL=HL+1
ldd  (HL),A      32         8 ----  (HL)=A, HL=HL-1
ldd  A,(HL)      3A         8 ----  A=(HL), HL=HL-1
```

GMB 16bit-Loadcommands

```
ld   rr,nn       x1 nn nn  12 ----  rr=nn (rr may be BC,DE,HL or SP)
ld   SP,HL       F9         8 ----  SP=HL
push rr          x5        16 ----  SP=SP-2  (SP)=rr   (rr may be BC,DE,HL,AF)
pop  rr          x1        12 (AF)  rr=(SP)  SP=SP+2   (rr may be BC,DE,HL,AF)
```

GMB 8bit-Arithmetic/logical Commands

```
add  A,r         8x         4 z0hc  A=A+r
add  A,n         C6 nn      8 z0hc  A=A+n
add  A,(HL)      86         8 z0hc  A=A+(HL)
adc  A,r         8x         4 z0hc  A=A+r+cy
adc  A,n         CE nn      8 z0hc  A=A+n+cy
adc  A,(HL)      8E         8 z0hc  A=A+(HL)+cy
sub  r           9x         4 z1hc  A=A-r
sub  n           D6 nn      8 z1hc  A=A-n
sub  (HL)        96         8 z1hc  A=A-(HL)
sbc  A,r         9x         4 z1hc  A=A-r-cy
sbc  A,n         DE nn      8 z1hc  A=A-n-cy
sbc  A,(HL)      9E         8 z1hc  A=A-(HL)-cy
and  r           Ax         4 z010  A=A & r
and  n           E6 nn      8 z010  A=A & n
and  (HL)        A6         8 z010  A=A & (HL)
xor  r           Ax         4 z000
xor  n           EE nn      8 z000
xor  (HL)        AE         8 z000
```

```
or   r          Bx         4 z000 A=A | r
or   n          F6 nn      8 z000 A=A | n
or   (HL)       B6         8 z000 A=A | (HL)
cp   r          Bx         4 z1hc compare A-r
cp   n          FE nn      8 z1hc compare A-n
cp   (HL)       BE         8 z1hc compare A-(HL)
inc  r          xx         4 z0h- r=r+1
inc  (HL)       34        12 z0h- (HL)=(HL)+1
dec  r          xx         4 z1h- r=r-1
dec  (HL)       35        12 z1h- (HL)=(HL)-1
daa             27         4 z-0x decimal adjust akku
cpl             2F         4 -11- A = A xor FF
```

GMB 16bit-Arithmetic/logical Commands

```
add  HL,rr     x9           8 -0hc HL = HL+rr     ;rr may be BC,DE,HL,SP
inc  rr        x3           8 ---- rr = rr+1      ;rr may be BC,DE,HL,SP
dec  rr        xB           8 ---- rr = rr-1      ;rr may be BC,DE,HL,SP
add  SP,dd     E8          16 00hc SP = SP +/- dd ;dd is 8bit signed number
ld   HL,SP+dd  F8          12 00hc HL = SP +/- dd ;dd is 8bit signed number
```

GMB Rotate- und Shift-Commands

```
rlca           07           4 000c rotate akku left
rla            17           4 000c rotate akku left through carry
rrca           0F           4 000c rotate akku right
rra            1F           4 000c rotate akku right through carry
rlc  r         CB 0x        8 z00c rotate left
rlc  (HL)      CB 06       16 z00c rotate left
rl   r         CB 1x        8 z00c rotate left through carry
rl   (HL)      CB 16       16 z00c rotate left through carry
rrc  r         CB 0x        8 z00c rotate right
rrc  (HL)      CB 0E       16 z00c rotate right
rr   r         CB 1x        8 z00c rotate right through carry
rr   (HL)      CB 1E       16 z00c rotate right through carry
sla  r         CB 2x        8 z00c shift left arithmetic (b0=0)
sla  (HL)      CB 26       16 z00c shift left arithmetic (b0=0)
swap r         CB 3x        8 z000 exchange low/hi-nibble
swap (HL)      CB 36       16 z000 exchange low/hi-nibble
sra  r         CB 2x        8 z00c shift right arithmetic (b7=b7)
sra  (HL)      CB 2E       16 z00c shift right arithmetic (b7=b7)
srl  r         CB 3x        8 z00c shift right logical (b7=0)
srl  (HL)      CB 3E       16 z00c shift right logical (b7=0)
```

GMB Singlebit Operation Commands

```
bit  n,r       CB xx        8 z01- test bit n
bit  n,(HL)    CB xx       12 z01- test bit n
set  n,r       CB xx        8 ---- set bit n
```

```
set  n,(HL)    CB xx       16 ---- set bit n
res  n,r       CB xx        8 ---- reset bit n
res  n,(HL)    CB xx       16 ---- reset bit n
```

GMB CPU-Controlcommands

```
ccf            3F           4 -00c cy=cy xor 1
scf            37           4 -001 cy=1
nop            00           4 ---- no operation
halt           76         N*4 ---- halt until interrupt occurs (low power)
stop           10 00        ? ---- low power standby mode (VERY low power)
di             F3           4 ---- disable interrupts, IME=0
ei             FB           4 ---- enable interrupts, IME=1
```

GMB Jumpcommands

```
jp   nn        C3 nn nn    16 ---- jump to nn, PC=nn
jp   HL        E9           4 ---- jump to HL, PC=HL
jp   f,nn      xx nn nn 16;12 ---- conditional jump if nz,z,nc,c
jr   PC+dd     18 dd       12 ---- relative jump to nn (PC=PC+/-7bit)
jr   f,PC+dd   xx dd     12;8 ---- conditional relative jump if nz,z,nc,c
call nn        CD nn nn    24 ---- call to nn, SP=SP-2, (SP)=PC, PC=nn
call f,nn      xx nn nn 24;12 ---- conditional call if nz,z,nc,c
ret            C9          16 ---- return, PC=(SP), SP=SP+2
ret  f         xx        20;8 ---- conditional return if nz,z,nc,c
reti           D9          16 ---- return and enable interrupts (IME=1)
rst  n         xx          16 ---- call to 00,08,10,18,20,28,30,38
```

CPU Comparision with Z80

Comparision with 8080

Basically, the gameboy CPU works more like an older 8080 CPU rather than like a more powerful Z80 CPU. It is, however, supporting CB-prefixed instructions. Also, all known gameboy assemblers using the more obvious Z80-style syntax, rather than the chaotic 8080-style syntax.

Comparision with Z80

Any DD-, ED-, and FD-prefixed instructions are missing, that means no IX-, IY-registers, no block commands, and some other missing commands.
All exchange instructions have been removed (including total absence of second register set), 16bit memory accesses are mostly missing, and 16bit arithmetic functions are heavily cut-down.
The gameboy has no IN/OUT instructions, instead I/O ports are accessed directly by normal LD instructions, or by special LD (FF00+n) opcodes.
The sign and parity/overflow flags have been removed.
The gameboy operates approximately as fast as a 4MHz Z80 (8MHz in CGB double speed mode), execution time of all instructions has been rounded up to a multiple of 4 cycles though.

Moved, Removed, and Added Opcodes

Opcode	Z80	GMB
08	EX AF,AF	LD (nn),SP
10	DJNZ PC+dd	STOP
22	LD (nn),HL	LDI (HL),A
2A	LD HL,(nn)	LDI A,(HL)
32	LD (nn),A	LDD (HL),A
3A	LD A,(nn)	LDD A,(HL)
D3	OUT (n),A	-
D9	EXX	RETI
DB	IN A,(n)	-
DD	<IX>	-
E0	RET PO	LD (FF00+n),A
E2	JP PO,nn	LD (FF00+C),A
E3	EX (SP),HL	-
E4	CALL P0,nn	-
E8	RET PE	ADD SP,dd
EA	JP PE,nn	LD (nn),A
EB	EX DE,HL	-
EC	CALL PE,nn	-
ED	<pref>	-
F0	RET P	LD A,(FF00+n)
F2	JP P,nn	LD A,(FF00+C)
F4	CALL P,nn	-
F8	RET M	LD HL,SP+dd
FA	JP M,nn	LD A,(nn)
FC	CALL M,nn	-
FD	<IY>	-
CB3X	SLL r/(HL)	SWAP r/(HL)

Note: The unused (-) opcodes will lock-up the gameboy CPU when used.

The Cartridge Header

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

0100-0103 - Entry Point
After displaying the Nintendo Logo, the built-in boot procedure jumps to this address (100h), which should then jump to the actual main program in the cartridge. Usually this 4 byte area contains a NOP instruction, followed by a JP 0150h instruction. But not always.

0104-0133 - Nintendo Logo

These bytes define the bitmap of the Nintendo logo that is displayed when the gameboy gets turned on. The hexdump of this bitmap is:

```
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
```

The gameboys boot procedure verifies the content of this bitmap (after it has displayed it), and LOCKS ITSELF UP if these bytes are incorrect. A CGB verifies only the first 18h bytes of the bitmap, but others (for example a pocket gameboy) verify all 30h bytes.

0134-0143 - Title

Title of the game in UPPER CASE ASCII. If it is less than 16 characters then the remaining bytes are filled with 00's. When inventing the CGB, Nintendo has reduced the length of this area to 15 characters, and some months later they had the fantastic idea to reduce it to 11 characters only. The new meaning of the ex-title bytes is described below.

013F-0142 - Manufacturer Code

In older cartridges this area has been part of the Title (see above), in newer cartridges this area contains an 4 character uppercase manufacturer code. Purpose and Deeper Meaning unknown.

0143 - CGB Flag

In older cartridges this byte has been part of the Title (see above). In CGB cartridges the upper bit is used to enable CGB functions. This is required, otherwise the CGB switches itself into Non-CGB-Mode. Typical values are:

```
80h - Game supports CGB functions, but works on old gameboys also.
C0h - Game works on CGB only (physically the same as 80h).
```

Values with Bit 7 set, and either Bit 2 or 3 set, will switch the gameboy into a special non-CGB-mode with uninitialized palettes. Purpose unknown, eventually this has been supposed to be used to colorize monochrome games that include fixed palette data at a special location in ROM.

0144-0145 - New Licensee Code

Specifies a two character ASCII licensee code, indicating the company or publisher of the game. These two bytes are used in newer games only (games that have been released after the SGB has been invented). Older games are using the header entry at 014B instead.

0146 - SGB Flag

Specifies whether the game supports SGB functions, common values are:

```
00h = No SGB functions (Normal Gameboy or CGB only game)
03h = Game supports SGB functions
```

The SGB disables its SGB functions if this byte is set to another value than 03h.

0147 - Cartridge Type

Specifies which Memory Bank Controller (if any) is used in the cartridge, and if further external hardware exists in the cartridge.

```
00h  ROM ONLY                 13h  MBC3+RAM+BATTERY
01h  MBC1                     15h  MBC4
02h  MBC1+RAM                 16h  MBC4+RAM
03h  MBC1+RAM+BATTERY         17h  MBC4+RAM+BATTERY
05h  MBC2                     19h  MBC5
```

```
06h  MBC2+BATTERY              1Ah  MBC5+RAM
08h  ROM+RAM                   1Bh  MBC5+RAM+BATTERY
09h  ROM+RAM+BATTERY           1Ch  MBC5+RUMBLE
0Bh  MMM01                     1Dh  MBC5+RUMBLE+RAM
0Ch  MMM01+RAM                 1Eh  MBC5+RUMBLE+RAM+BATTERY
0Dh  MMM01+RAM+BATTERY         FCh  POCKET CAMERA
0Fh  MBC3+TIMER+BATTERY        FDh  BANDAI TAMA5
10h  MBC3+TIMER+RAM+BATTERY    FEh  HuC3
11h  MBC3                      FFh  HuC1+RAM+BATTERY
12h  MBC3+RAM
```

0148 - ROM Size

Specifies the ROM Size of the cartridge. Typically calculated as "32KB shl N".

```
00h -  32KByte (no ROM banking)
01h -  64KByte (4 banks)
02h - 128KByte (8 banks)
03h - 256KByte (16 banks)
04h - 512KByte (32 banks)
05h -   1MByte (64 banks)  - only 63 banks used by MBC1
06h -   2MByte (128 banks) - only 125 banks used by MBC1
07h -   4MByte (256 banks)
52h - 1.1MByte (72 banks)
53h - 1.2MByte (80 banks)
54h - 1.5MByte (96 banks)
```

0149 - RAM Size

Specifies the size of the external RAM in the cartridge (if any).

```
00h - None
01h - 2 KBytes
02h - 8 Kbytes
03h - 32 KBytes (4 banks of 8KBytes each)
```

When using a MBC2 chip 00h must be specified in this entry, even though the MBC2 includes a built-in RAM of 512 x 4 bits.

014A - Destination Code

Specifies if this version of the game is supposed to be sold in japan, or anywhere else. Only two values are defined.

```
00h - Japanese
01h - Non-Japanese
```

014B - Old Licensee Code

Specifies the games company/publisher code in range 00-FFh. A value of 33h signalizes that the New License Code in header bytes 0144-0145 is used instead.
(Super GameBoy functions won't work if <> $33.)

014C - Mask ROM Version number

Specifies the version number of the game. That is usually 00h.

014D - Header Checksum
Contains an 8 bit checksum across the cartridge header bytes 0134-014C. The checksum is calculated as follows:

```
x=0:FOR i=0134h TO 014Ch:x=x-MEM[i]-1:NEXT
```

The lower 8 bits of the result must be the same than the value in this entry. The GAME WON'T WORK if this checksum is incorrect.

014E-014F - Global Checksum
Contains a 16 bit checksum (upper byte first) across the whole cartridge ROM. Produced by adding all bytes of the cartridge (except for the two checksum bytes). The Gameboy doesn't verify this checksum.

Memory Bank Controllers

As the gameboys 16 bit address bus offers only limited space for ROM and RAM addressing, many games are using Memory Bank Controllers (MBCs) to expand the available address space by bank switching. These MBC chips are located in the game cartridge (ie. not in the gameboy itself), several different MBC types are available:

In each cartridge, the required (or preferred) MBC type should be specified in byte at 0147h of the ROM. (As described in the chapter about The Cartridge Header.)

None (32KByte ROM only)

Small games of not more than 32KBytes ROM do not require a MBC chip for ROM banking. The ROM is directly mapped to memory at 0000-7FFFh. Optionally up to 8KByte of RAM could be connected at A000-BFFF, even though that could require a tiny MBC-like circuit, but no real MBC chip.

MBC1 (max 2MByte ROM and/or 32KByte RAM)

This is the first MBC chip for the gameboy. Any newer MBC chips are working similiar, so that is relative easy to upgrade a program from one MBC chip to another - or even to make it compatible to several different types of MBCs.

Note that the memory in range 0000-7FFF is used for both reading from ROM, and for writing to the MBCs Control Registers.

0000-3FFF - ROM Bank 00 (Read Only)
This area always contains the first 16KBytes of the cartridge ROM.

4000-7FFF - ROM Bank 01-7F (Read Only)
This area may contain any of the further 16KByte banks of the ROM, allowing to address up to 125 ROM Banks (almost 2MByte). As described below, bank numbers 20h, 40h, and 60h cannot be used, resulting in the odd amount of 125 banks.

A000-BFFF - RAM Bank 00-03, if any (Read/Write)
This area is used to address external RAM in the cartridge (if any). External RAM is often battery buffered, allowing to store game positions or high score tables, even if the gameboy is turned off, or if the cartridge is removed from the gameboy. Available RAM sizes are: 2KByte (at A000-A7FF), 8KByte (at A000-BFFF), and 32KByte (in form of four 8K banks at A000-BFFF).

0000-1FFF - RAM Enable (Write Only)
Before external RAM can be read or written, it must be enabled by writing to this address space. It is recommended to disable external RAM after accessing it, in order to protect its contents from damage during power down of the gameboy. Usually the following values are used:

```
00h  Disable RAM (default)
0Ah  Enable RAM
```

Practically any value with 0Ah in the lower 4 bits enables RAM, and any other value disables RAM.

2000-3FFF - ROM Bank Number (Write Only)
Writing to this address space selects the lower 5 bits of the ROM Bank Number (in range 01-1Fh). When 00h is written, the MBC translates that to bank 01h also. That doesn't harm so far, because ROM Bank 00h can be always directly accessed by reading from 0000-3FFF.
But (when using the register below to specify the upper ROM Bank bits), the same happens for Bank 20h, 40h, and 60h. Any attempt to address these ROM Banks will select Bank 21h, 41h, and 61h instead.

4000-5FFF - RAM Bank Number - or - Upper Bits of ROM Bank Number (Write Only) This 2bit register can be used to select a RAM Bank in range from 00-03h, or to specify the upper two bits (Bit 5-6) of the ROM Bank number, depending on the current ROM/RAM Mode. (See below.)

6000-7FFF - ROM/RAM Mode Select (Write Only)
This 1bit Register selects whether the two bits of the above register should be used as upper two bits of the ROM Bank, or as RAM Bank Number.

```
00h = ROM Banking Mode (up to 8KByte RAM, 2MByte ROM) (default)
01h = RAM Banking Mode (up to 32KByte RAM, 512KByte ROM)
```

The program may freely switch between both modes, the only limitiation is that only RAM Bank 00h can be used during Mode 0, and only ROM Banks 00-1Fh can be used during Mode 1.

MBC2 (max 256KByte ROM and 512x4 bits RAM)

0000-3FFF - ROM Bank 00 (Read Only)
Same as for MBC1.

4000-7FFF - ROM Bank 01-0F (Read Only)
Same as for MBC1, but only a total of 16 ROM banks is supported.

A000-A1FF - 512x4bits RAM, built-in into the MBC2 chip (Read/Write)
The MBC2 doesn't support external RAM, instead it includes 512x4 bits of built-in RAM (in the MBC2 chip itself). It still requires an external battery to save data during power-off though.
As the data consists of 4bit values, only the lower 4 bits of the "bytes" in this memory area are used.

0000-1FFF - RAM Enable (Write Only)
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.

2000-3FFF - ROM Bank Number (Write Only)
Writing a value (XXXXBBBB - X = Don't cares, B = bank select bits) into 2000-3FFF area will select an appropriate ROM bank at 4000-7FFF.

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 (max 2MByte ROM and/or 32KByte RAM and Timer)

Beside for the ability to access up to 2MB ROM (128 banks), and 32KB RAM (4 banks), the MBC3 also includes a built-in Real Time Clock (RTC). The RTC requires an external 32.768 kHz Quartz Oscillator, and an external battery (if it should continue to tick when the gameboy is turned off).

0000-3FFF - ROM Bank 00 (Read Only)
Same as for MBC1.

4000-7FFF - ROM Bank 01-7F (Read Only)
Same as for MBC1, except that accessing banks 20h, 40h, and 60h is supported now.

A000-BFFF - RAM Bank 00-03, if any (Read/Write)
A000-BFFF - RTC Register 08-0C (Read/Write)
Depending on the current Bank Number/RTC Register selection (see below), this memory space is used to access an 8KByte external RAM Bank, or a single RTC Register.

0000-1FFF - RAM and Timer Enable (Write Only)
Mostly the same as for MBC1, a value of 0Ah will enable reading and writing to external RAM - and to the RTC Registers! A value of 00h will disable either.

2000-3FFF - ROM Bank Number (Write Only)
Same as for MBC1, except that the whole 7 bits of the RAM Bank Number are written directly to this address. As for the MBC1, writing a value of 00h, will select Bank 01h instead. All other values 01-7Fh select the corresponding ROM Banks.

4000-5FFF - RAM Bank Number - or - RTC Register Select (Write Only)
As for the MBC1s RAM Banking Mode, writing a value in range for 00h-03h maps the corresponding external RAM Bank (if any) into memory at A000-BFFF.
When writing a value of 08h-0Ch, this will map the corresponding RTC register into memory at A000-BFFF. That register could then be read/written by accessing any address in that area, typically that is done by using address A000.

6000-7FFF - Latch Clock Data (Write Only)
When writing 00h, and then 01h to this register, the current time becomes latched into the RTC registers. The latched data will not change until it becomes latched again, by repeating the write 00h->01h procedure.
This is supposed for <reading> from the RTC registers. It is proof to read the latched (frozen) time from the RTC registers, while the clock itself continues to tick in background.

The Clock Counter Registers

```
08h  RTC S   Seconds   0-59 (0-3Bh)
09h  RTC M   Minutes   0-59 (0-3Bh)
0Ah  RTC H   Hours     0-23 (0-17h)
0Bh  RTC DL  Lower 8 bits of Day Counter (0-FFh)
0Ch  RTC DH  Upper 1 bit of Day Counter, Carry Bit, Halt Flag
      Bit 0  Most significant bit of Day Counter (Bit 8)
      Bit 6  Halt (0=Active, 1=Stop Timer)
      Bit 7  Day Counter Carry Bit (1=Counter Overflow)
```

The Halt Flag is supposed to be set before <writing> to the RTC Registers.

The Day Counter
The total 9 bits of the Day Counter allow to count days in range from 0-511 (0-1FFh). The Day Counter Carry Bit becomes set when this value overflows. In that case the Carry Bit remains set until the program does reset it.
Note that you can store an offset to the Day Counter in battery RAM. For example, every time you read a non-zero Day Counter, add this Counter to the offset in RAM, and reset the Counter to zero. This method allows to count any number of days, making your program Year-10000-Proof, provided that the cartridge gets used at least every 511 days.

Delays
When accessing the RTC Registers it is recommended to execute a 4ms delay (4 Cycles in Normal Speed Mode) between the separate accesses.

HuC1 (MBC with 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. (Similiar to the infrared port that has been later invented in CGBs.)

The Japanese cart "Fighting Phoenix" (internal cart name: SUPER B DAMAN) is known to contain this chip.

MBC Timing Issues

Using MBCs with CGB Double Speed Mode
The MBC5 has been designed to support CGB Double Speed Mode.
There have been rumours that older MBCs (like MBC1-3) wouldn't be fast enough in that mode. If so, it might be nethertheless possible to use Double Speed during periods which use only code and data which is located in internal RAM.
However, despite of the above, my own good old selfmade MBC1-EPROM card appears to work stable and fine even in Double Speed Mode though.

Gamegenie/Shark Cheats

Game Shark and Gamegenie are external cartridge adapters that can be plugged between the gameboy and the actual game cartridge. Hexadecimal codes can be then entered for specific games, typically providing things like Infinite Sex, 255 Cigarettes, or Starting directly in Wonderland Level PRO, etc.

Gamegenie (ROM patches)
Gamegenie codes consist of nine-digit hex numbers, formatted as ABC-DEF-GHI, the meaning of the separate digits is:

```
AB    New data
FCDE  Memory address, XORed by 0F000h
GI    Old data, XORed by 0BAh and rotated left by two
H     Don't know, maybe checksum and/or else
```

The address should be located in ROM area 0000h-7FFFh, the adapter permanently compares address/old data with address/data being read by the game, and replaces that data by new data if necessary. That method (more or less) prohibits unwanted patching of wrong memory banks. Eventually it is also possible to patch external RAM ?
Newer devices reportedly allow to specify only the first six digits (optionally). As far as I rememeber, around three or four codes can be used

simultaneously.

Game Shark (RAM patches)
Game Shark codes consist of eight-digit hex numbers, formatted as ABCDEFGH, the meaning of the separate digits is:

```
AB    External RAM bank number
CD    New Data
GHEF  Memory Address (internal or external RAM, A000-DFFF)
```

As far as I understand, patching is implement by hooking the original VBlank interrupt handler, and re-writing RAM values each frame. The downside is that this method steals some CPU time, also, it cannot be used to patch program code in ROM.
As far as I rememeber, somewhat 10-25 codes can be used simultaneously.

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. If all locations compare the same, then 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. If it is a zero, then the internal ROM is disabled and cartridge program execution begins at location $100 with the following register values:

```
AF=$01B0
BC=$0013
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.

Reducing Power Consumption

The following can be used to recude the power consumption of the gameboy, and to extend the life of the batteries.

PWR Using the HALT Instruction

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.

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.

When waiting for a vblank event, this would be a BAD example:

```
@@wait:
 ld   a,(0FF44h)      ;LY
 cp   a,144
 jr   nz,@@wait
```

A better example would be a procedure as shown below. In this case the vblank interrupt must be enabled, and your vblank interrupt procedure must set vblank_flag to a non-zero value.

```
 ld   hl,vblank_flag  ;hl=pointer to vblank_flag
 xor  a               ;a=0
@@wait:               ;wait...
 halt                 ;suspend CPU - wait for ANY interrupt
 cp   a,(hl)          ;vblank flag still zero?
 jr   z,@@wait        ;wait more if zero
 ld   (hl),a          ;set vblank_flag back to zero
```

The vblank_flag is used to determine whether the HALT period has been terminated by a vblank interrupt, or by another interrupt. In case that your program has all other interrupts disabled, then it would be proof to replace the above procedure by a single HALT instruction.

PWR Using the STOP Instruction

The STOP instruction is intended to switch the gameboy into VERY low power standby mode. For example, a program may use this feature when it hasn't sensed keyboard input for a longer period (assuming that somebody forgot to turn off the gameboy).

Before invoking STOP, it might be required to disable Sound and Video manually (as well as IR-link port in CGB). Much like HALT, the STOP state is terminated by interrupt events - in this case this would be commonly a joypad interrupt. The joypad register might be required to be prepared for STOP either.

PWR Disabeling the Sound Controller

If your programs doesn't use sound at all (or during some periods) then write 00h to register FF26 to save 16% or more on GB power consumption. Sound can be turned back on by writing 80h to the same register, all sound registers must be then re-initialized.

When the gameboy becomes turned on, sound is enabled by default, and must be turned off manually when not used.

PWR Not using CGB Double Speed Mode

Because CGB Double Speed mode consumes more power, it'd be recommended to use normal speed when possible.
There's limited ability to switch between both speeds, for example, a game might use normal speed in the title screen, and double speed in the game, or vice versa.
However, during speed switch the display collapses for a short moment, so that it'd be no good idea to alter speeds within active game or title screen periods.

PWR Using the Skills

Most of the above power saving methods will produce best results when using efficient and tight assembler code which requires as less CPU power as possible. Thus, experienced old-school programmers will (hopefully) produce lower power consumption, as than HLL-programming teenagers, for example.

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 rr        dec rr          ;rr = bc,de, or hl
ldi a,(hl)    ldd a,(hl)
ldi (hl),a    ldd (hl),a
```

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

External Connectors

Cartridge Slot

```
Pin   Name    Expl.
1     VDD     Power Supply +5V DC
2     PHI     System Clock
3     /WR     Write
4     /RD     Read
5     /CS     Chip Select
6-21  A0-A15  Address Lines
```

```
22-29 D0-D7   Data Lines
30    /RES    Reset signal
31    VIN     External Sound Input
32    GND     Ground
```

Link Port

Pin numbers are arranged as 2,4,6 in upper row, 1,3,5 un lower row; outside view of gameboy socket; flat side of socket upside.
Colors as used in most or all standard link cables, because SIN and SOUT are crossed, colors Red and Orange are exchanged at one cable end.

```
Pin Name Color  Expl.
1   VCC  -      +5V DC
2   SOUT red    Data Out
3   SIN  orange Data In
4   P14  -      Not used
5   SCK  green  Shift Clock
6   GND  blue   Ground
```

Note: The original gameboy used larger plugs (unlike pocket gameboys and newer), linking between older/newer gameboys is possible by using cables with one large and one small plug though.

Stereo Sound Connector (3.5mm, female)

```
Pin     Expl.
Tip     Sound Left
Middle  Sound Right
Base    Ground
```

External Power Supply

...

END