CHAPTER 1: SYSTEM

1. GENERAL SYSTEM INFORMATION

1.1 System Overview

Structure

At the heart of the DMG/CGB system is a CPU with a built-in LCD controller designed for DMG/CGB use.

System

[DMG]	[CGB]
➢ Dot-matrix LCD unit capable of grayscale display	➢ Color dot-matrix LCD unit capable of RGB with 32 grayscale shades
➢ 64 Kbit – SRAM (for LCD display)	➢ 128 Kbit – SRAM (for LCD display)
➢ 64 Kbit – SRAM (working memory)	➢ 256 Kbit – SRAM (working memory)
	➢ Infrared communication link (photo transistor, photo LED)

Features common to DMG/CGB

- 32-pin connector (for ROM cartridge connection)
- 6-pin subconnector (for external serial communication)
- DC-DC converter for power source
- Sound amp
- Keys for operation
- Speaker
- Stereo headphone connector
- Input connector for external power source

Types of Game Pak Supported

1 Game Boy Game Pak
(Software that uses only the Game Boy functions. When used with Game Boy Color, 4-10 colors are displayed.)

2 Game Boy Color Game Pak

- Game Pak supported by CGB (for use with both CGB and DMG)
- Game Pak for CGB only (software that runs only on CGB)

Operating Modes (the following modes apply only to CGB)

1 DMG Mode (when using software for DMG)
The new registers, expanded memory area, and new features for CGB are not used. Color applications previously associated with palette data BGP, OBP0, and OBP1 are performed by the system.

2 CGB Mode (when using software supported or used exclusively by CGB)
The new registers, expanded memory area, and new features of CGB are available.

Note	***To operate in CGB mode, specific code must first be placed in the ROM data area of the user program. For more information, see Chapter 5, Section 2,* Recognition of CGB support (CGB only) in ROM Data.**

Power Source

- Battery/AC adapter/Battery charger

Accessories (as of April 1999)

DMG Accessories

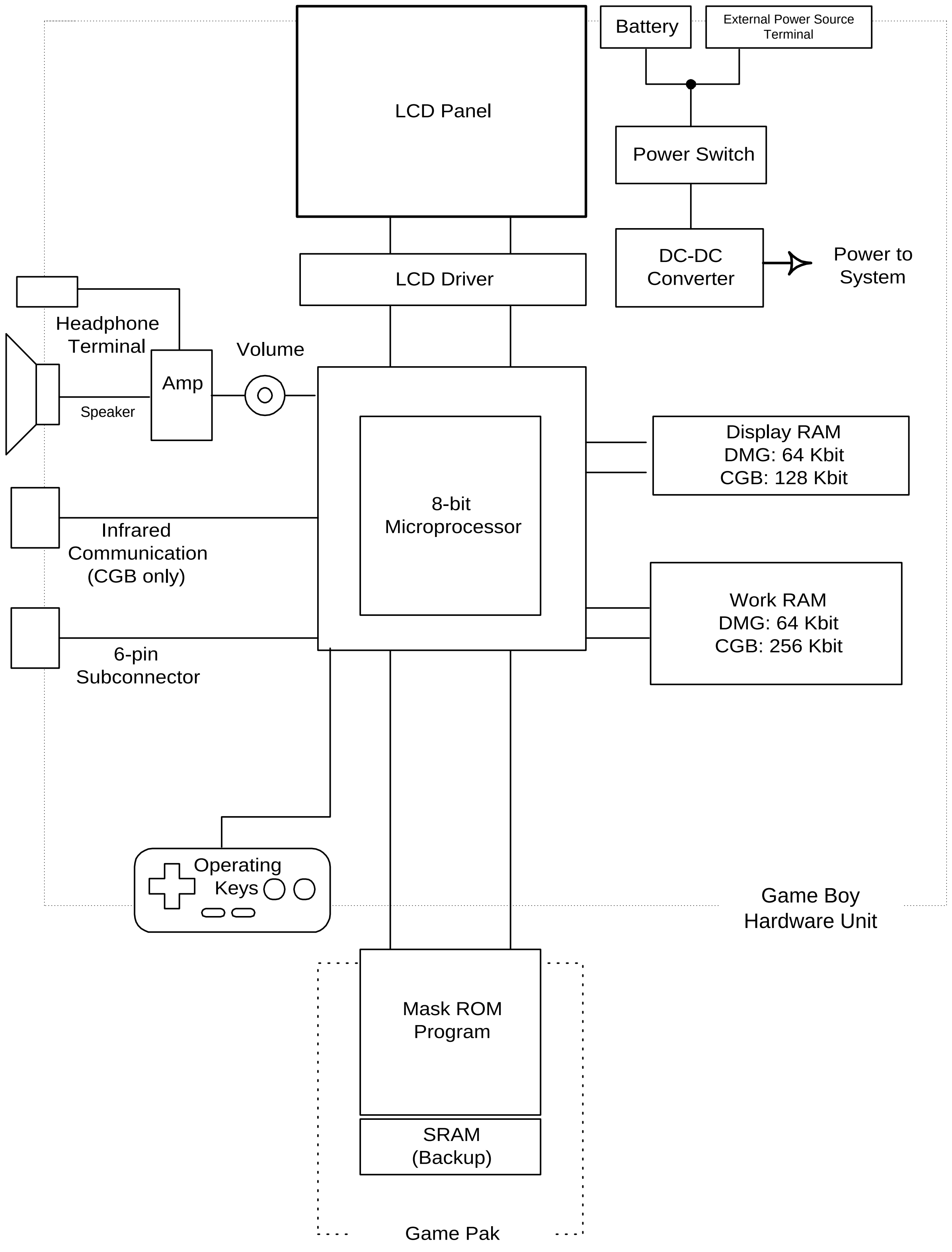
- Communication Cable
- Battery Charger Adapter

MGB/CGB Accessories

- Communication Cable
- AC Adapter
- Battery Pack Charger Set

The 6-pin serial communication subconnector and the AC adapter input connector of the DMG hardware that preceded MGB are shaped differently than those of MGB and CGB. Thus, two types of accessories are available — those exclusively for DMG and those exclusively for MGB/CGB. In addition, a conversion connector is necessary for communication between DMG and MGB/CGB.

1.2 GAME BOY BLOCK DIAGRAM



1.3 MEMORY CONFIGURATION

In DMG and CGB, the 32 KB from 0x0 to 0x7FFF is available as program area.

0x000-0x0FF: Allocated as the destination address for RST instructions and the starting address for interrupts.
0x100-0x14F: Allocated as the ROM area for storing data such as the name of the game.
0x150: Allocated as the starting address of the user program.

The 8 KB from 0x8000 to 0x9FFF is used as RAM for the LCD display. In CGB, the amount of RAM allocated for this purpose is 16 KB (8 KB x 2), twice the amount allocated for the LCD display in DMG, and this RAM can be used in 8 KB units using bank switching. The 8 KB RAM areas are divided into the following 2 areas.

1 An area for character data
2 An area for BG (background) display data (Character code and attribute)

The 8 KB from 0xA000 to 0xBFFF is the area allocated for external expansion RAM.
The 8 KB from 0xC000 to 0xDFFF is the work RAM area.
In DMG, the 8 KB of working RAM is implemented without change. In CGB, bank switching is used to provide 32 KB of working RAM. This 32 KB area is divided into 8 areas of 4 KB each.

1 The 4 KB from 0xC000 to 0xCFFF is fixed as Bank 0.
2 The 4 KB from 0xD000 to 0xDFFF can be switched between banks 1 though 7.

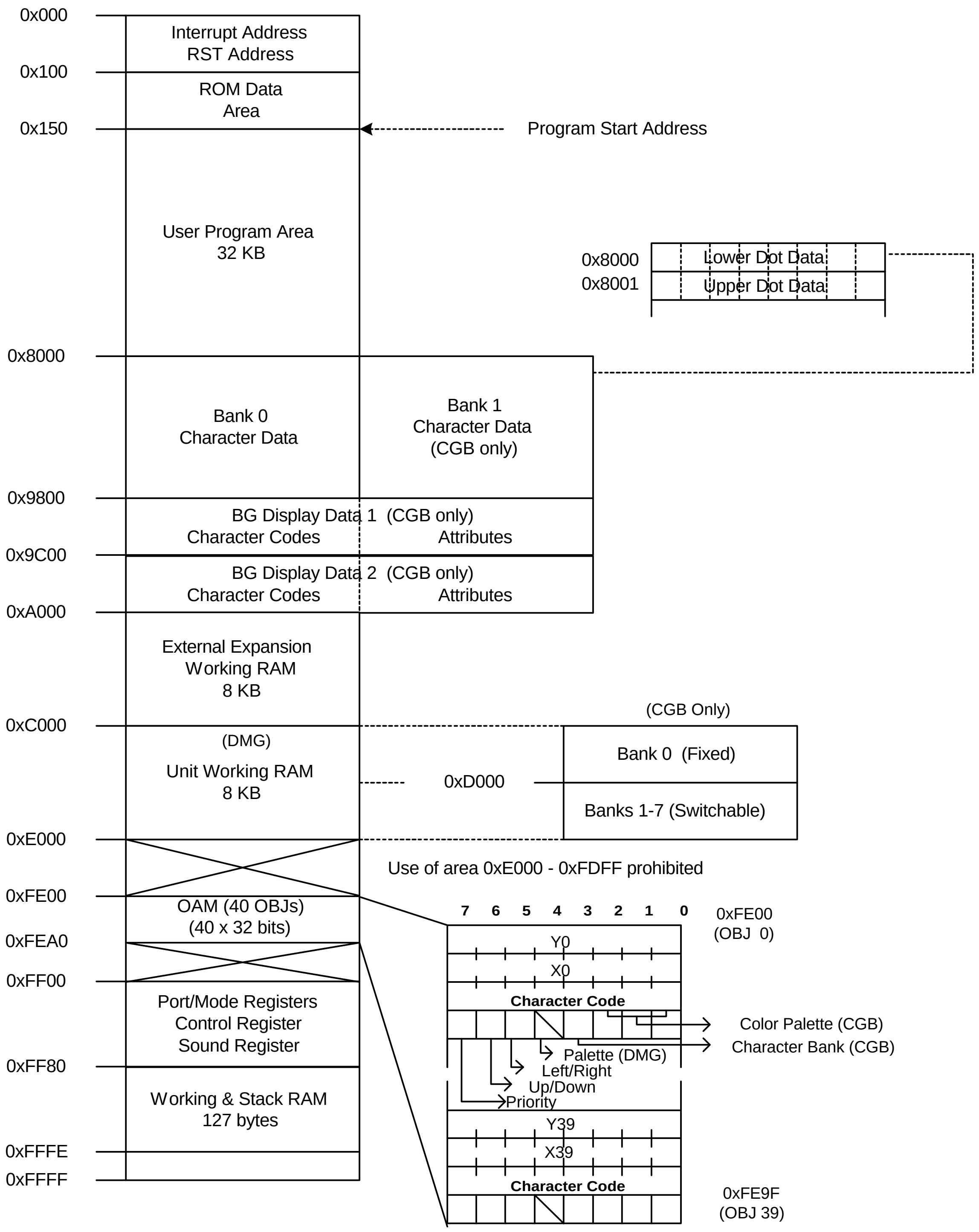
Note Use of the area from 0xE000 to 0xFDFF is prohibited.

0xFE00 to 0xFFFF is allocated for CPU internal RAM.

0xFE00-0xFE9F: OAM-RAM (Holds display data for 40 objects)
0xFF00-0xFF7F & 0xFFFF: Specified for purposes such as instruction registers and system controller flags.
0xFF80-0xFFFE: Can be used as CPU work RAM and/or stack RAM.

1.4 MEMORY MAP

Note	***In DMG, there is no bank switching at 0x8000-0x9FFF and 0xC000-0xDFFF.***



1.5 FEATURE COMPARISON

Item	**DMG CPU**	**CGB CPU**
CPU Speed (system operating frequency)	1.05 MHz	1.05 MHz (normal mode) 2.10 MHz (double-speed mode)
Game Boy RAM		
Work and Stack RAM	127 x 8 bits	←
Work RAM	8,192 bytes	32,768 bytes
OAM	40 x 28 bits	40 x 32 bits
For LCD display	8,192 bytes	16,384 bytes
Game Pak Memory Space		
ROM (without MBC)	32,768 bytes	←
RAM (without MBC)	8,192 bytes	←
LCD Controller		
Display Capacity	160 x 144 dots	160 x 144 x RGB dots
Block Structure		
BG, window	8 x 8 dots	←
Object	8 x 8 dots or 8 x 16 dots	←
Number of Usable Characters		
BG	256	512
OBJ 8 x 8	256	512
8 x 16	128	256
Grayscale: BG, window	4 shades, 1 palette	4 colors, 8 palettes (DMG mode: 4 colors, 1 palette)
Grayscale: Object	3 shades, 2 palettes	3 colors, 8 palettes (DMG mode: 3 colors, 2 palettes)
Object priority		
Different x coordinates	Object with smallest x coord .	Object with lowest OBJ number (DMG mode: Object with lowest x coord.)
Same x coordinates	Object with lowest OBJ number	←
Timer & Divider Stages	8-bit timer x 1 16 stages x 1	← ←
Serial Input/Output	8 bits x 1	←
Baud Rate	8 K	8K/256K (16K/512K in high-speed mode
DMA Controller		
Existing DMA	0x8000~0xDFFF→OAM	0x0~0xDFFF→OAM
Horizontal blank DMA	---	Game Pak & Work RAM→VRAM
General-purpose DMA	---	Game Pak & Work RAM→VRAM
Interrupt features		
Internal Interrupts	4 types (maskable)	←
External Interrupts	1 type (maskable)	←
Input/Output Ports		
Serial Input/Output Ports	SIN, SCK, SOUT	←
Infrared Communication Port	---	R0, R1, R2, R3
Sound Output Circuit	4 sounds	← Monaural (VIN) External Sound Mixable Input

←: Same as in column at left

1.6 REGISTER COMPARISON

	DMG CPU		**CGB CPU**	
Use	**Register**	**Address**	**Register**	**Address**
Port/Mode Registers	P1	FF00	←	←
	SB	FF01	←	←
	SC	FF02	←	←
	DIV	FF04	←	←
	TIMA	FF05	←	←
	TMA	FF06	←	←
	TAC	FF07	←	←
		---	KEY1	FF4D
		---	RP	FF56
Bank Control Registers		---	VBK	FF4F
		---	SVBK	FF70
Interrupt Flags	IF	FF0F	←	←
	IE	FFFF	←	←
	IME		←	
LCD Display Registers	LCDC	FF40	←	←
	STAT	FF41	←	←
	SCY	FF42	←	←
	SCX	FF43	←	←
	LY	FF44	←	←
	LYC	FF45	←	←
	DMA	FF46	←	←
	BGP	FF47	←	←
	OBP0	FF48	←	←
	OBP1	FF49	←	←
	WY	FF4A	←	←
	WX	FF4B	←	←
		---	HDMA1	FF51
		---	HDMA2	FF52
		---	HDMA3	FF53
		---	HDMA4	FF54
		---	HDMA5	FF55
		---	BCPS	FF68
		---	BCPD	FF69
		---	OCPS	FF6A
		---	OCPD	FF6B
	OAM	FE00~FE9F	←	←
Sound Registers	NR x x	FF10~FF26	←	←
	Waveform RAM	FF30~FF3F	←	←

←: Same as in column at left

2. CPU

2.1 OVERVIEW OF CPU FEATURES

The CPUs of DMG and CGB are ICs customized for DMG/CGB use, and have the following features.

CPU Features

Central to the 8-bit CPU are the following features, including an I/O port and timer.

- 127 x 8 bits of built-in RAM (working and stack)
- RAM for LCD Display: <DMG> 8 KB/<CGB>16 KB ( )
- Working RAM: <DMG> 8KB/<CGB> 32 KB
- Built-in 16-stage Frequency Divider
- Built-in 8-bit Timer
- 4 types of Internal Interrupts (maskable)
- 1 type of External Interrupt (maskable)
- Built-in DMA Controller
- Input Ports P10 ~ P13
- Output Ports P14 and P15
- Serial I/O Ports SIN, SCK, SOUT
- Infrared I/O Port <CGB only>

LCD Controller Functions

Game Boy is equipped with functions that provide control of the images displayed on the LCD. Character data used for display is held in system RAM.

- DMG: 4 shades of gray; CGB: 32 shades of gray for each RGB color
- 160 x 144-dot liquid crystal display
- 8 x 8-dot composition of background and window characters
- 8 x 8 or 8 x 16-dot composition of OBJ characters
- Up to 40 objects displayable in 1 screen
- Up to 10 objects displayable on 1 horizontal line
- 40 x 32 bits of built-in RAM (OBJ-RAM for LCD)
- Control of 256 x 256-dot background
- Vertically and horizontally scrollable background
- Window-like functions

Sound Functions

Each system is equipped with 4 types of sound synthesis circuitry.

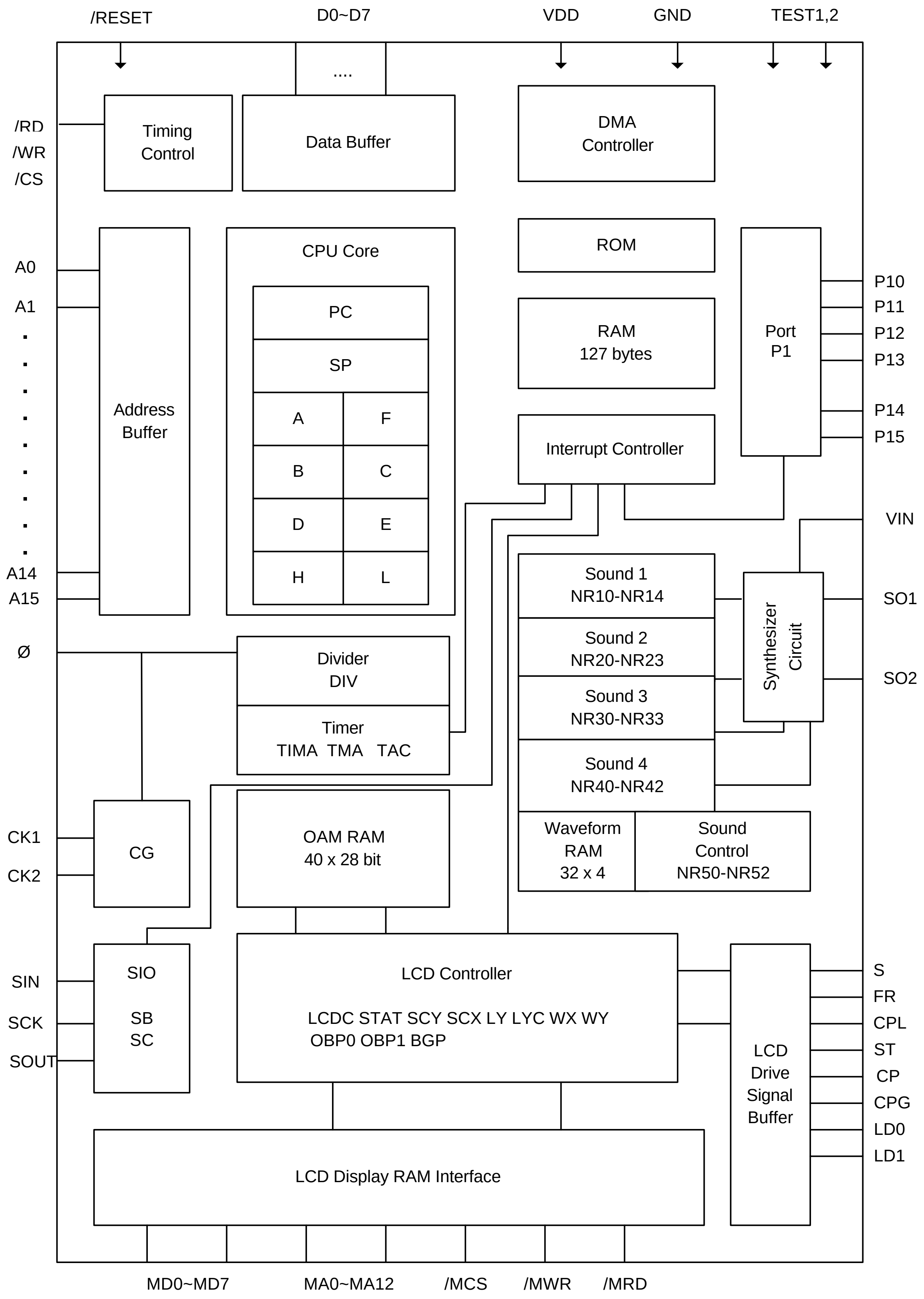
- Sound 1: Quadrangular waveform, sweep and envelope functions
- Sound 2: Quadrangular waveform, envelope functions
- Sound 3: Arbitrary waveform, generated
- Sound 4: White noise, generated
- 2 output channels (output can be allocated to a channel)
- Synthesized output with external sound input <CGB only>

Miscellaneous

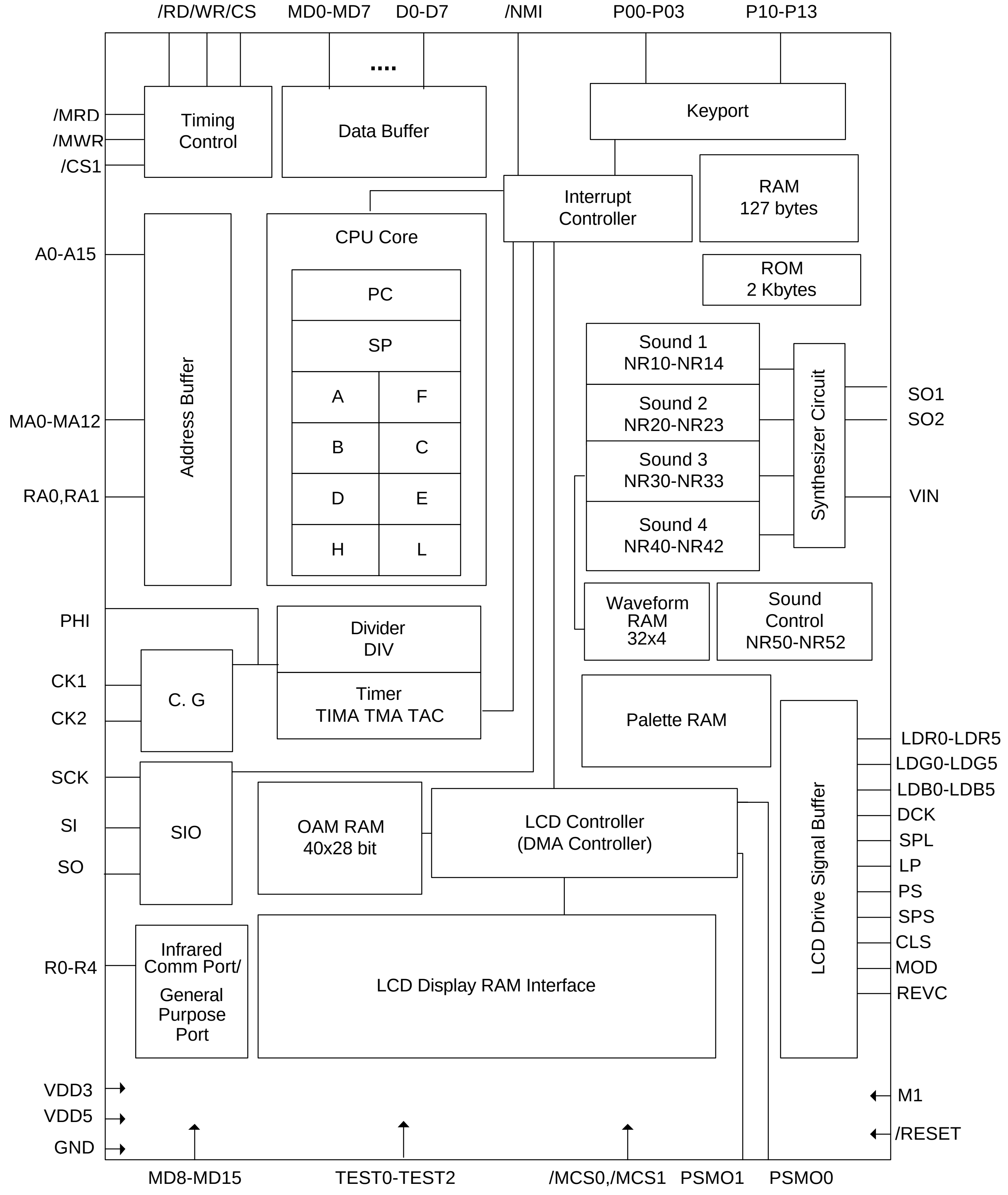
- An internal monitor program is built into DMG/CGB CPUs. When power is turned on or the Game Boy is reset, the internal monitor program first initializes components such as the ports, then passes control to the user program.
- Instruction cycles
  <DMG> 0.954 μs (source oscillation: 4.1943 MHz)
  <CGB> 0.954 μs/0.477 μs, switchable (source oscillation: 8.3886 MHz)

2.2 CPU BLOCK DIAGRAM

Game Boy (DMG/MGB) CPU



Game Boy Color CPU



2.3 DESCRIPTION OF CPU FUNCTIONS

Interrupts

There are five types of interrupts available, including 4 types of maskable internal interrupts and 1 type of maskable external interrupt. The IE flag is used to control interrupts. The IF flag indicates which type of interrupt is set.

- ➢ LCD Display Vertical Blanking
- ➢ Status Interrupts from LCDC (4 modes)
- ➢ Timer Overflow Interrupt
- ➢ Serial Transfer Completion Interrupt
- ➢ End of Input Signal for ports P10-P13

DMA Transfers

DMA transfers are controlled by the DMA registers.
<DMG>
DMG allows 40 x 32-bit DMA transfers from 0x8000-0xDFFF to OAM (0xFE00-0xFE9F). The transfer start address can be specified in increments of 0x100 for 0x8000-0xDFFF.
<CGB>
In addition to the DMA transfers method for DMG (from 0x0000-0xDFFF in CGB), CGB enables two new types of DMA transfer — horizontal blanking and general-purpose DMA transfers.
Note, however, that when performing a DMG-type DMA transfer on CGB, some consideration must be given to specifying the destination RAM area.
For more information, see the DMA Functions section in Chapter 2.

1 Horizontal Blanking DMA Transfer
   Sixteen bytes of data are automatically transferred for each horizontal blanking period during a DMA transfer from the user program area (0x0000-0x7FFF) or external and hardware working RAM area (0xA000-0xDFFF) to the LCD display RAM area (0x8000-0x9FFF).

2 General-Purpose DMA Transfer
   Between 16 and 2048 bytes of data (specified in 16-byte increments) are transferred from the user program area (0x0000-0x7FFF) or external and hardware working RAM area (0xA000-0xDFFF) to the LCD display RAM area (0x8000-0x9FFF), during the Vertical Blanking Period.

Timer

The timer is composed of the following:

- ➢ TIMA (timer counter)
- ➢ TMA (timer modulo register)
- ➢ TAC (timer control register)

Controller Connections

- ➢ P10-P13: Input ports
- ➢ P14-P15: The key matrix structure is composed of the output ports.

At user program startup, the status of the CPU port registers and mode registers are as follows.

Register	Status
P1	0
SC	0
TIMA	0
TAC	0
IE	0
LCDC	$83 BG/OBJ ON, LCDC OPERATION
SCY	0
SCX	0
LYC	0
WY	0
W	0
Interrupt Enable (IE)	DI

Stack: 0xFFFE

Standby Modes

The standby functions are HALT mode, which halts the system clock, and STOP mode, which halts oscillation (source oscillation).

HALT Mode

Game Boy switches to HALT mode when a HALT instruction is executed.
The system clock and CPU operation halt in this mode. However, operation of source oscillation circuitry between terminals CK1 and CK2 continues. Thus, the functions that do not require the system clock (e.g,, DIV, SIO, timer, LCD controller, and sound circuit) continue to operate in this mode.
HALT mode is canceled by the following events, which have the starting addresses indicated.

1) A LOW signal to the /RESET terminal
   Starting address: 0x0000
2) The interrupt-enable flag and its corresponding interrupt request flag are set
   IME = 0 (Interrupt Master Enable flag disabled)
   Starting address: address following that of the HALT instruction
   IME = 1 (Interrupt Master Enable flag enabled)
   Starting address: each interrupt starting address

STOP Mode

Game Boy switches to STOP mode when a STOP instruction is executed.
The system clock and oscillation circuitry between the CK1 and CK2 terminals are halted in this mode. Thus, all operation is halted except that of the SI0 external clock. STOP mode is canceled by the following events, and started from the starting address.

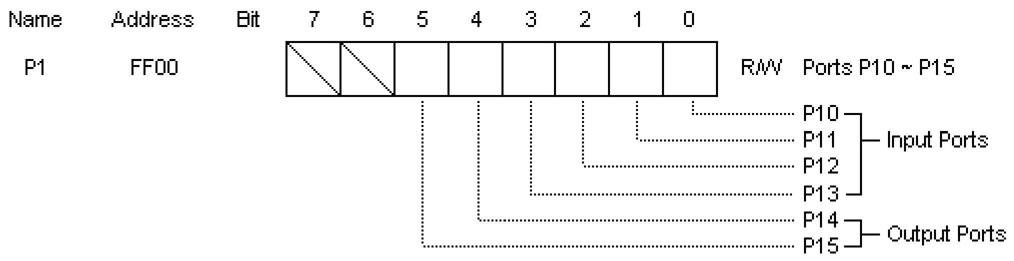
3) A LOW signal to the /RESET terminal
   Starting address: 0x0000
4) A LOW signal to terminal P10, P11, P12, or P13
   Starting address: address following that of STOP instruction

When STOP mode is canceled, the system clock is restored after 2^{17} times the oscillation clock (DMG: 4 MHz, CGB: 4 MHz/8 MHz), and the CPU resumes operation.
When STOP mode is entered, the STOP instruction should be executed after all interrupt-enable flags are reset, and meanwhile, terminals P10-P13 are all in a HIGH period.

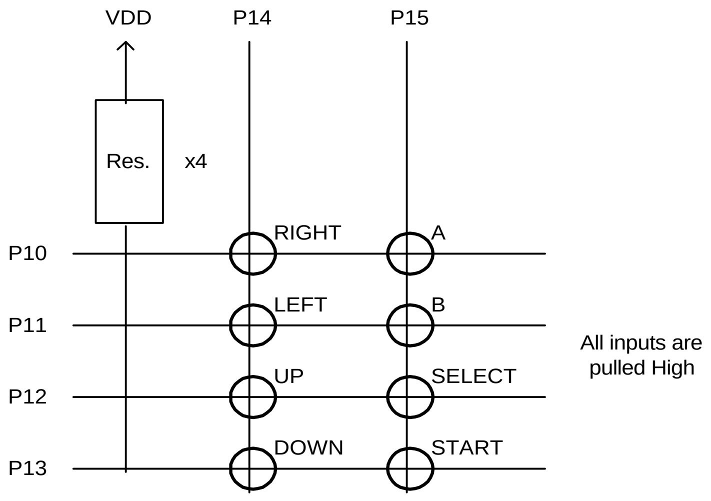
2.4 CPU FUNCTIONS (COMMON TO DMG/CGB①)

The CPU functions described here are those that are identical in DMG and CGB. CPU functions that are enhanced in CGB are described in Section 2.5, *CPU Functions (Common to DMG/CGB②)*. CPU functions that cannot be used for DMG are described in Section 2.6, *CPU Function (CGB only)*.

2.4.1 Controller Data



The P1 ports are connected with a matrix for reading key operations.



When key input is read, a brief interval is interposed between P14 and P15 output and reading of the input, as shown below.

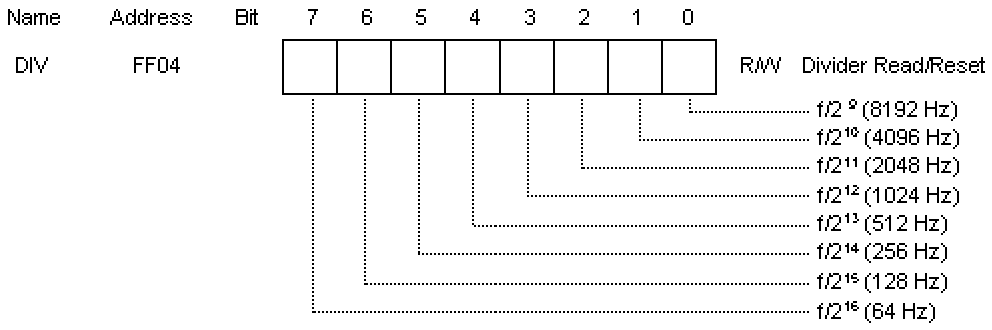
```
Example:  KEY    LD       A, $20          ; Read U, D, L, R keys
                 LD       ($FF00), A      ; Port P14  ← LOW output
                 LD       A, ($FF00)      ; Register A ← Port P10-P13
                 LD       A, ($FF00)      ; Perform this operation twice
                    .
                    .
                 LD       A, ($10)        ; Reads keys A, B, SE, ST
                 LD       ($FF00), A      ; Port P15 ← LOW output

                 LD       A, ($FF00)      ; Register A ← Ports P10-P13
                 LD       A, ($FF00)      ; Perform this operation 6 times
                 LD       A, ($FF00)      ;
                    .                     ;
                    .                     ;
                 LD       A, $30          ; Port reset
                 LD       ($FF00), A
                    .
                 RET
```

The interrupt request flag (IF: 4) is set by negative edge input at one of the P13-P10 terminals. Negative edge input requires a LOW period of 2^4 times source oscillation (DMG = 4 MHz, CGB = 4 MHz/8 MHz).

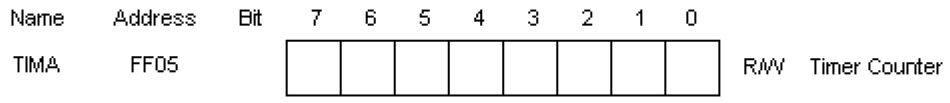
The interrupt request flag (IF: 4) also is set when a reset signal is input to the /RESET terminal with a P13~P10 terminal in the LOW state.

2.4.2 Divider Registers

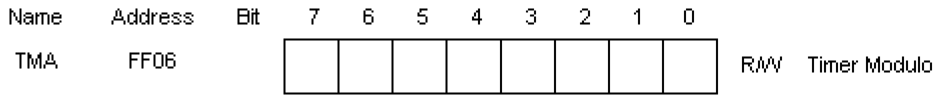


The upper 8 bits of the 16-bit counter that counts the basic clock frequency (f) can be referenced. If an LD instruction is executed, these bits are cleared to 0 regardless of the value being written. f = (4.194304 MHz).

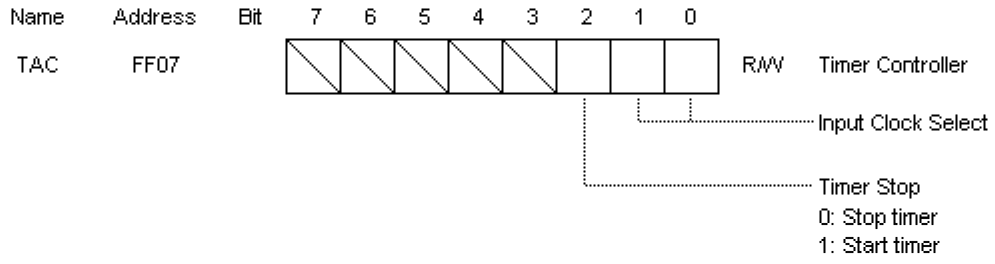
2.4.3 Timer Registers



The main timer unit. Generates an interrupt when it overflows.



The value of TMA is loaded when TIMA overflows.



Input Clock Select
00: f/2^{10} (4.096 KHz)
01: f/2^4 (262.144 KHz)
10: f/2^6 (65.536 KHz)
11: f/2^8 (16.384 KHz)

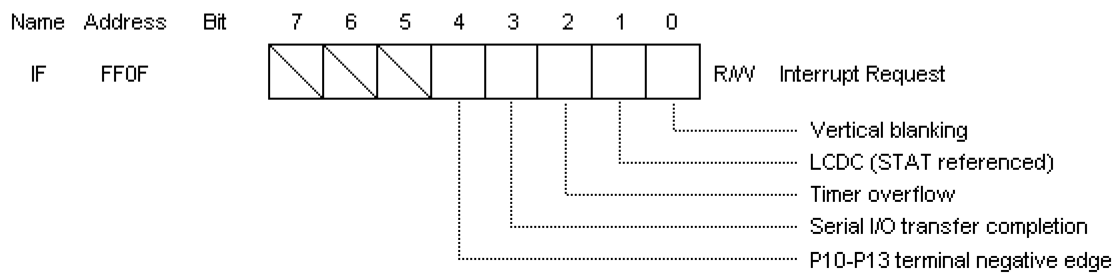
The timer consists of TIMA, TMA, and TAC.
The timer input clock is selected by TAC.
TIMA is the timer itself and operates using the clock selected by TAC.
TMA is the modulo register of TIMA. When TIMA overflows, the TMA data is loaded into TIMA.
Writing 1 to the 2nd bit of TAC starts the timer.
The timer should be started (the TAC start flag set) after the count up pulse is selected. Starting the timer before or at the same time as the count up pulse is selected may result in excessive count up operation.

Example

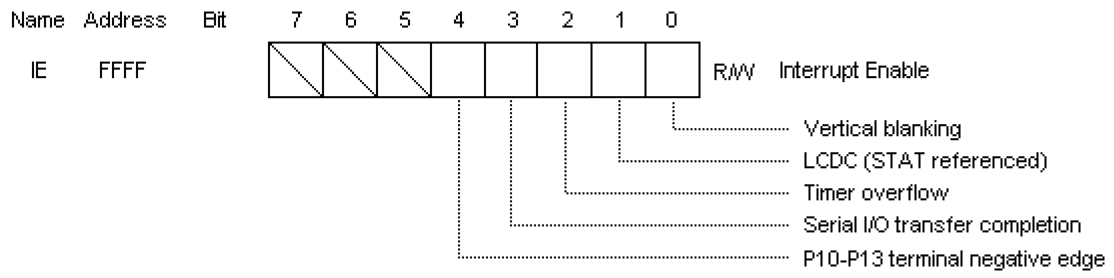
```
L D       A, 3          ;Select a count pulse of f/2^8
L D       (07), A       ;TAC ← 3 set
L D       A, 7          ;Start timer
L D       (07), A       ;
```

If a TMA write is executed with the same timing as that with which the contents of the modula register TMA are transferred to TIMA as the result of a timer overflow, the same data is transferred to TIMA.

2.4.4 Interrupt Flags



Bit reset enabled



0: Disabled
1: Enabled

Name	
IME	Interrupt Master Enable 0: Reset by DI instruction; prohibits all interrupts. 1: Set by EI instruction; the interrupt set by the IE registers are enabled.

Bit reset enabled

Interrupts are controlled by the IE (interrupt enable) flag.
The IF (interrupt request) flag can be used to determine which interrupt was requested.

The 5 types of interrupts are as follows.

Cause of Interrupt	Priority	Interrupt starting address
Vertical blanking	1	0x0040
LCDC status interrupt	2	0x0048
Timer overflow	3	0x0050
Serial transfer completion	4	0x0058
P10-P13 input signal goes low	5	0x0060

The LCDC interrupt mode can be selected (see STAT register)

- Mode 00
- Mode 01
- Mode 10
- LYC=LY consist

When multiple interrupts occur simultaneously, the IE flag of each is set, but only that with the highest priority is started. Those with lower priorities are suspended.

When using an interrupt, set the IF register to 0 before setting the IE register.

The interrupt process is as follows:

1 When an interrupt is processed, the corresponding IF flag is set.
2 Interrupt enabled.
  If the IME flag (Interrupt Master Enable) and the corresponding IE flag are set, the interrupt is performed by the following steps.
3 The IME flag is reset, and all interrupts are prohibited.
4 The contents of the PC (program counter) are pushed onto the stack RAM.
5 Control jumps to the interrupt starting address of the interrupt.

The resetting of the IF register that initiates the interrupt is a hardware reset.

The interrupt processing routine should push the registers during interrupt processing.

When an interrupt begins, all other interrupts are prohibited, but processing of the highest level interrupt is enabled by controlling the IME and IE flags with instructions.

Return from the interrupt routine is performed by the RET1 and RET instructions.

If the RETI instruction is used for the return, the IME flag is automatically set even if a DI instruction is executed in the interrupt processing routine.

IF the RET instruction is used for the return, the IME flag remains reset unless an EI instruction is executed in the interrupt routine.

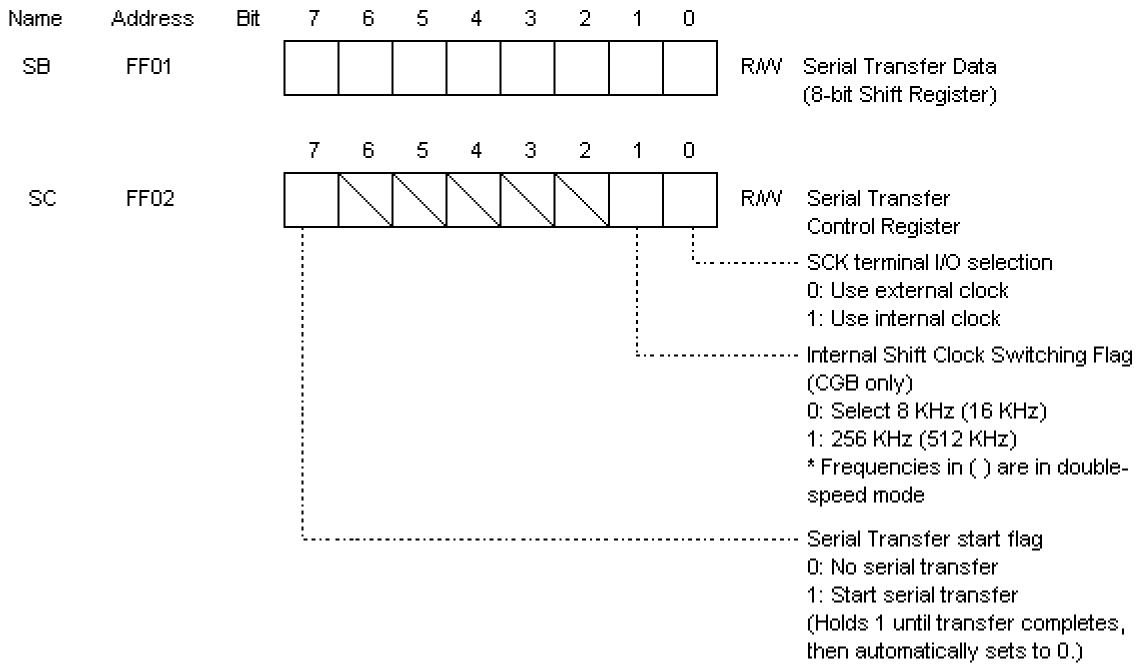
Each interrupt request flag of the IF register can be individually tested using instructions.

Interrupts are accepted during the op code fetch cycle of each instruction.

2.5 CPU FUNCTIONS (COMMON TO DMG/CGB②)

This section describes the CPU functions that have been enhanced in CGB. Functions that are identical in DMG and CGB are described in Section 2.4, *CPU Functions (Common to DMG/CGB①)*. CPU functions not available in DMG are described in Section 2.6, *CPU Functions (CGB only)*.

2.5.1 Serial Cable Communication



Note	***In DMG mode, bit 1 of the SC register is set to 1 and cannot be changed, but the transfer speed is fixed at 8 KHz.***

Serial I/O (SIO) is controlled by the SB and SC registers.
The lowest bit (SC0) of the SC register can be used to select shift clock to be either the external clock from the SCK terminal or the internal shift clock.
Sending and receiving occur simultaneously with a serial transfer.
If the data to be sent is set in the SB register and the serial transfer is then started, the received data is set in the SB register when the transfer is finished.

Serial transfer procedure:

1. The data is set in the SB register.
2. Setting the highest SC register bit (SC 7) to 1 starts the transfer.
3. The 3-bit counter is reset and after 8 counts of the shift clock, the transfer is performed until overflow occurs.
4. SC7 is reset.
5. If the serial transfer completion interrupt is enabled, the CPU is interrupted.

When the shift clock goes low, the contents of the SB register are shifted leftward and the data is output from the highest bit. When the shift clock goes high, input data from the SIN terminal are output to the lowest bit of the SB register.

When the SCK terminal is in external-clock mode, it is pulled up to VDD.

If the highest bit of the SC register (SC7) is set, reading and writing to the SB register is prohibited.

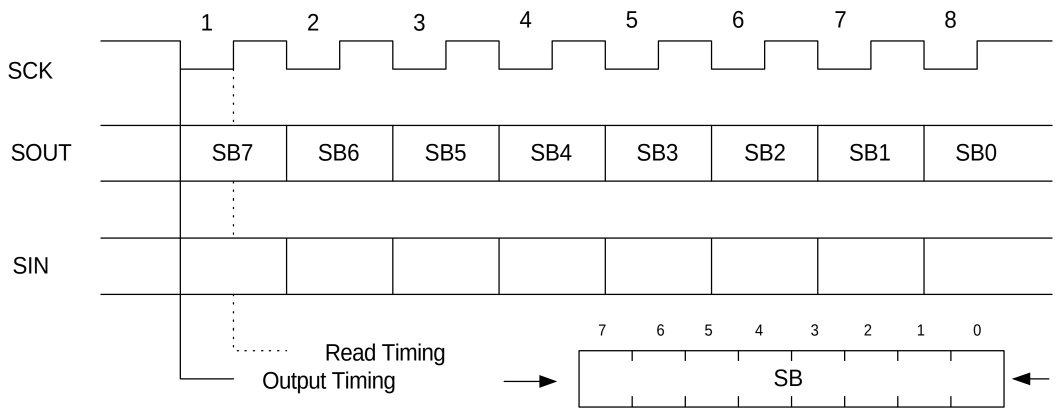
An SIO serial transfer should be started (highest SC bit set) after the external or internal shift clock is selected. Excessive shifting may result if the transfer is started before or at the same time as the shift clock is selected.

If a transfer is performed using the external clock, the data is first set in the SB register, then the SC register start flag is set and input from the external clock is awaited. The transfer start flag must be set each time data is transferred.

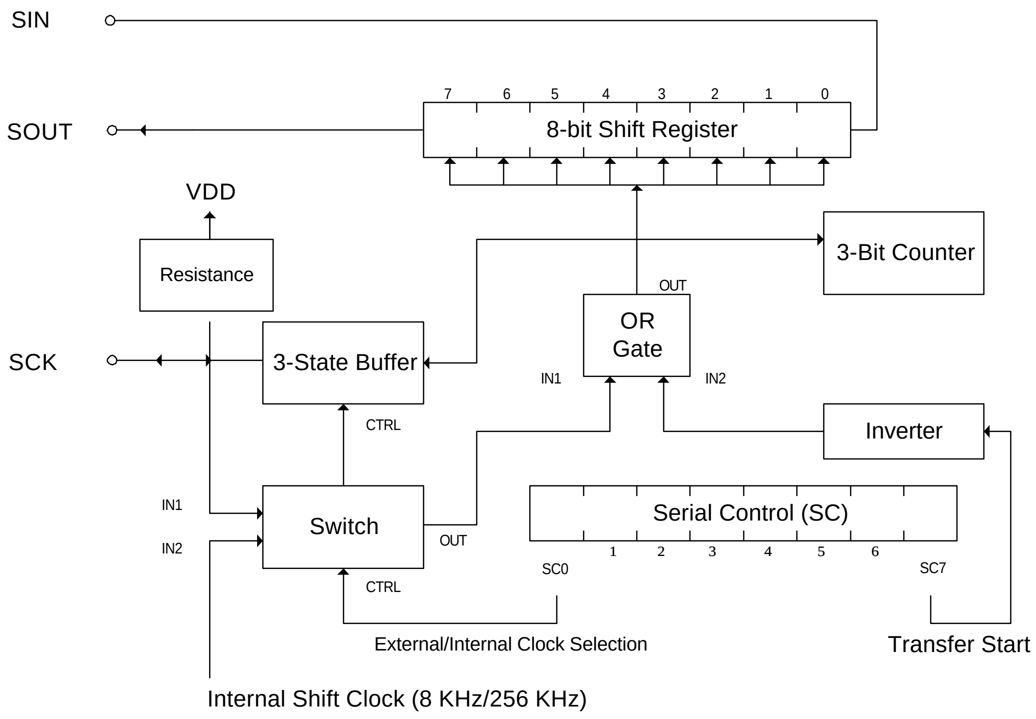
The maximum setting for an external clock is 500 KHz.

Serial communication (SIO) specifications are essentially the same for DMG and CGB. In CGB, however, the operating speed of the internal shift clock can be set to high by specifying a speed in bit 1.

SIO Timing Chart

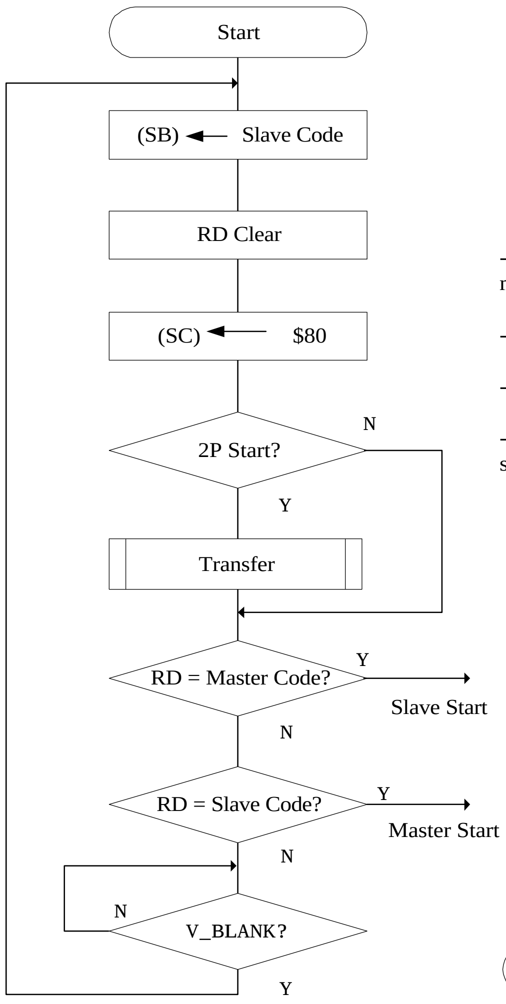


SIO Block Diagram



2.5.2 Serial Cable Communication: Reference flowchart

Flow until start of game



-Select code other than $00 and $FF. (For both slave and master code).

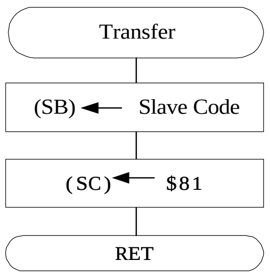
-Clear the receive data buffer (RD).

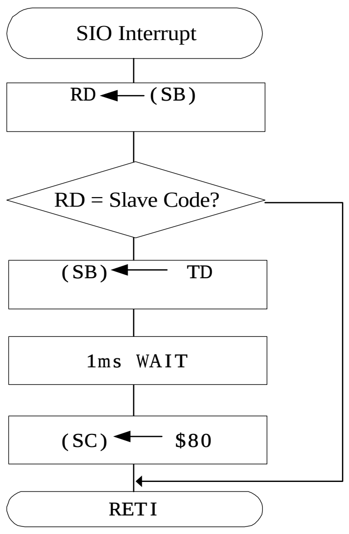
-Both sides wait in receive-wait status.

-Game on which Start key pressed first becomes master by sending master code to other game.

-Game first notified that it is slave by master code sent from master. Subsequently moves to game flow.

-Data sent when this side becomes master is the slave code. Game subsequently moves to game flow.

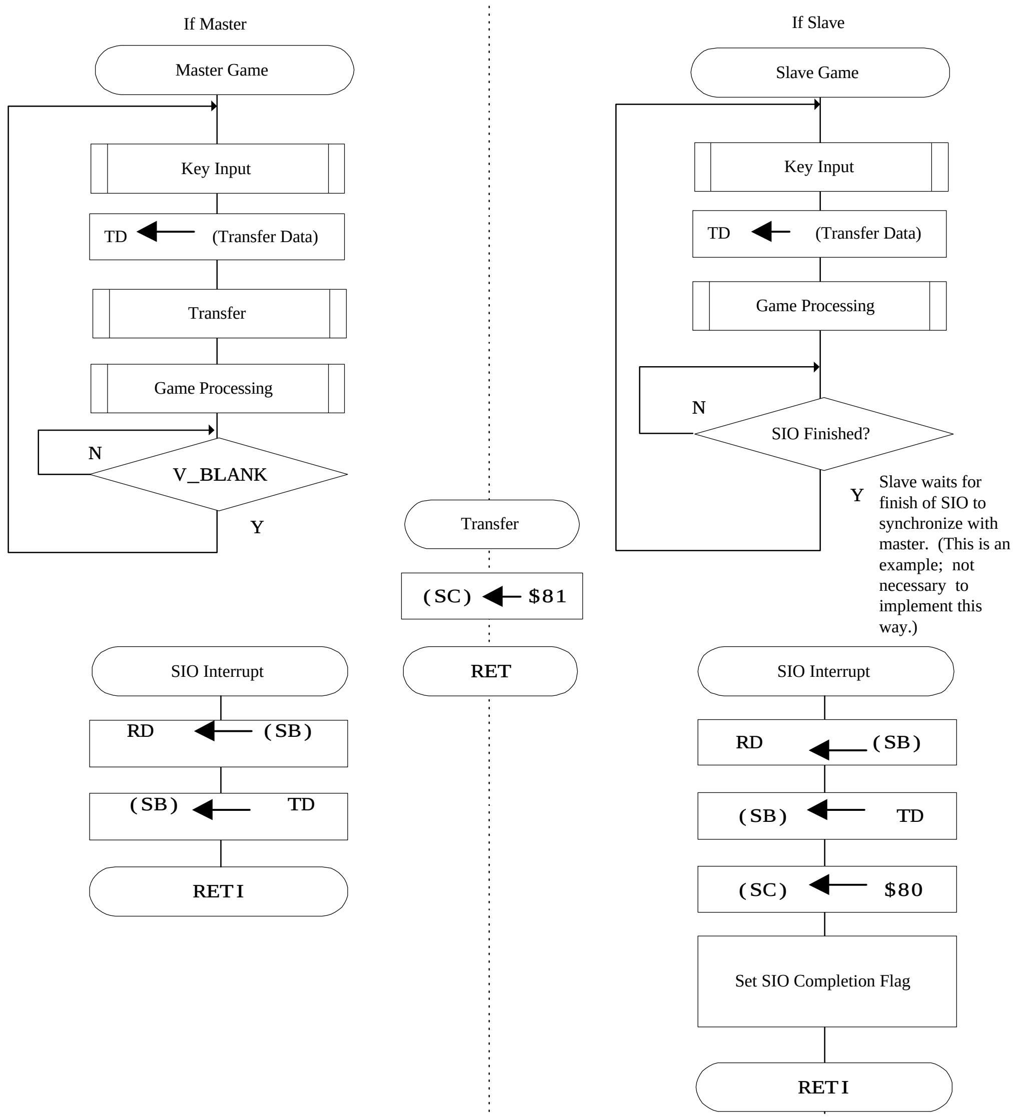




TD: Transfer Data Buffer

Timing of receive synchronized with Power Up.

Flow after game start



Data subsequently sent by the master is placed in (SB) and then sent to the slave at the same time as the (SC) is set to $81. At exactly that same time, the master receives the slave data. An SIO interrupt is then set in the slave and, as the flowchart indicates, the slave sets the data to be sent to the master (current data).

Because the data sent from the slave are those loaded at the time of the previous interrupt, the data sent to the master are one step (one pass through the main program) behind the current slave data. Exactly the converse is true when this process is viewed from the perspective of the slave. An SIO interrupt is set in the master, and the master sets the data to be sent to the slave (current data). In this case, because the data sent from the master are those loaded at the time of the previous interrupt, the data sent to slave are one step (one pass through main program) behind the current master data. (*The data of the master and slave can be synchronized by setting the data for each back 1 pass.)

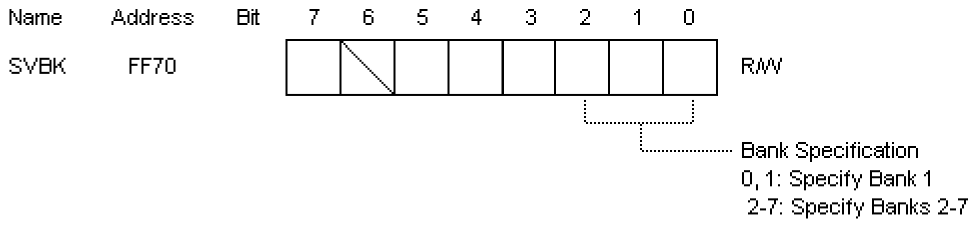
In the example, 1 byte is sent per frame. (This is not required.) If several bytes are sent continuously, a transmission interval longer than the processing time of other interrupts (e.g. V_BLANK) should be used (usually around 1 mS). The reason is that if an attempt is made to communicate with the slave during another interrupt, the slave cannot receive the data until after the interrupt is finished. If the next data is transmitted before the other interrupt is finished, the slave will be unable to receive the initial data of the transmission.

2.6 CPU FUNCTIONS (CGB ONLY)

This section describes CPU functions that can be used only with CGB. Functions that are identical in DMG and CGB are described in Section 2.4, *CPU Functions (Common to DMG/CGB①)*. For information on CPU functions enhanced in CGB, see Section 2.5, *CPU Functions (Common to DMG/CGB②)*.

2.6.1 Bank Register for Game Boy Working RAM

The 32 KB of Game Boy working RAM is divided into 8 banks of 4 KB each. The CPU memory space 0xC000-0xCFFF is set to Bank 0, and the space 0xD000-0xDFFF is switched between banks 1-7. Switching is performed using the lowest 3 bits of the bank register, SVBK. (If 0 is specified, Bank 1 is selected.)



Note This register cannot be written to in DMG mode.

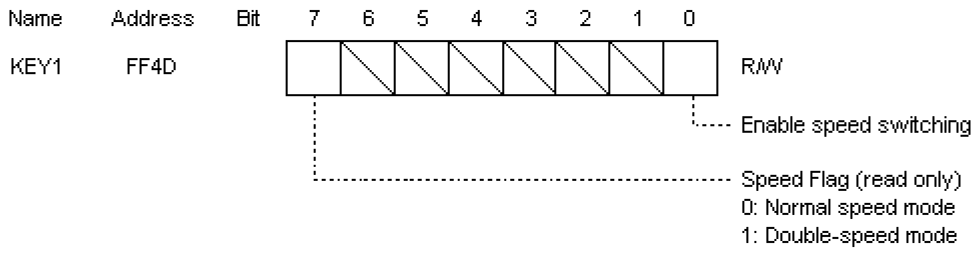
2.6.2 CPU Operating Speed

The speed of the CGB CPU can be changed to suit different purposes. In normal mode, each block operates at the same speed as with the DMG CPU. In double-speed mode, all blocks except the liquid crystal control circuit and the sound circuit operate at twice normal speed.

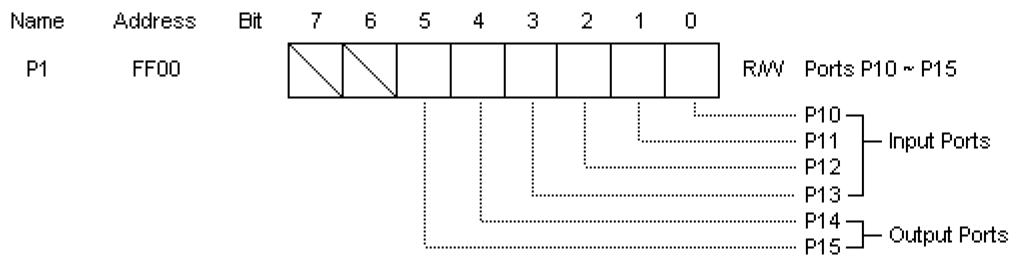
Normal mode: 1.05 MHz (CPU system clock)
Double-speed mode: 2.10 MHz (CPU system clock)

◆ Switching the CPU Operating Speed

Immediately after the CGB CPU is reset (immediately after reset cancellation), it operates in normal mode. The CPU mode is switched by executing a STOP instruction with bit 0 of register Key 1 set to a value of 1. If this is done in normal mode, the CPU is switched to double-speed mode; otherwise it is switched to normal mode. Bit 0 of register Key 1 is automatically reset after the operating speed is switched. In addition, bit 7 of register Key 1 serves as the CPU speed flag, indicating the current CPU speed.



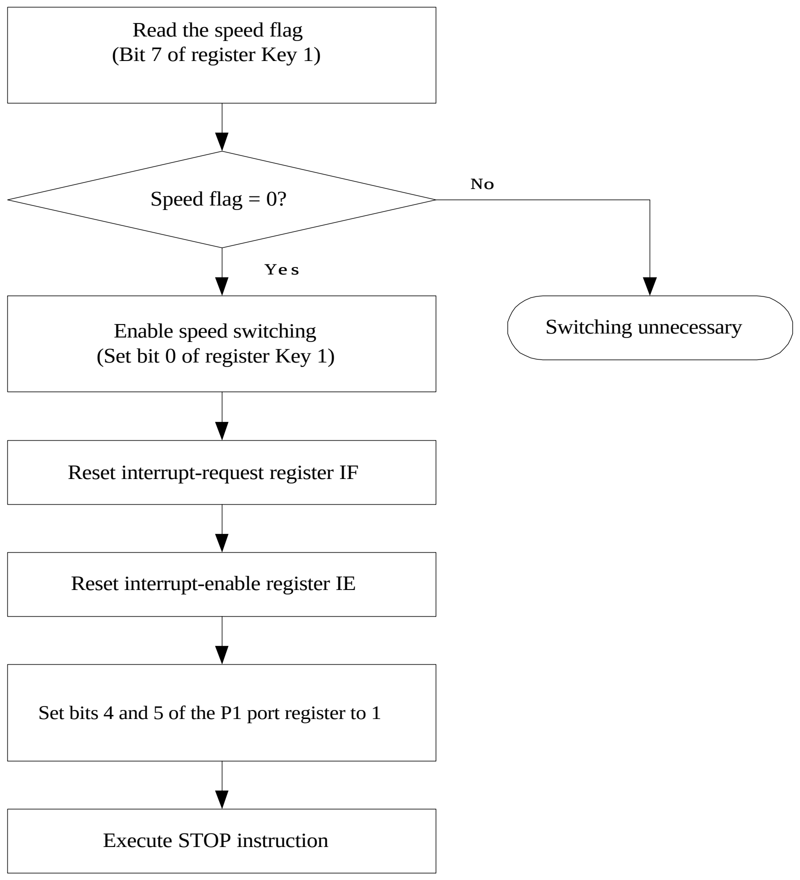
Note ***When bit 0 of register Key 1 is set to 1, the standby function cannot be used. When using the standby function, always confirm that bit 0 of register Key 1 is set to 0. When switching the CPU speed, all interrupt-enable flags should be reset and a STOP instruction executed with bits 4 and 5 of the P1 port register set to 1, as with the standby function (STOP mode). When the CPU speed is switched, a return from STOP mode is automatic, so it is not necessary to generate a STOP mode cancellation. However, until the CPU speed has been changed and the system clock returns, bits 4 and 5 of the P1 port register should be made to hold the value 1.***



Approximately 16 ms is required to switch from normal to double-speed mode, and approximately 32 ms is needed to switch from double-speed to normal mode. In double-speed mode, the DIV register (0xFF04) and the TIMA register (0xFF05) both operate at double speed. Battery life is shorter in double-speed mode than in normal mode. <u>The use of double-speed mode requires the corresponding mask ROM and MBC</u>.

Flow of Switching (when switching to double-speed mode)

In case the CPU operating speed needed to be switched, the current speed should always be checked first using the speed flag (bit 7 of the KEY 1 register). This ensures that the speed will be switched to the intended speed.


Read the speed flag
(Bit 7 of register Key 1)
Speed flag = 0?
No
Yes
Enable speed switching
(Set bit 0 of register Key 1)
Switching unnecessary
Reset interrupt-request register IF
Reset interrupt-enable register IE
Set bits 4 and 5 of the P1 port register to 1
Execute STOP instruction

Switching Routine (example)

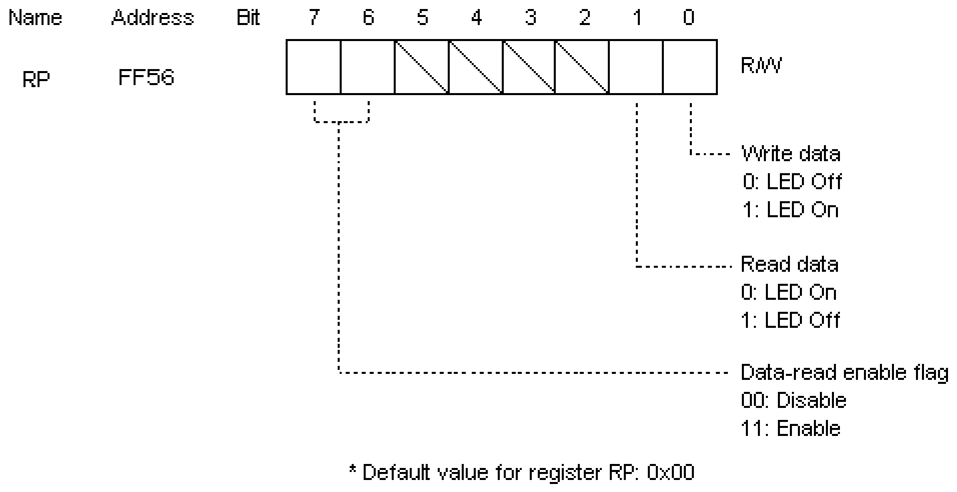
```
        LD      HL, KEY1
        BIT     7, (HL)
        JR      NZ, _NEXT
        SET     0, (HL)
        XOR     A
        LD      (IF), A
        LD      (IE), A
        LD      A, $30
        LD      (P1), A
        STOP
_NEXT
```

2.6.3 Infrared Communication

2.6.3.1 Port Register

The CGB system is equipped with an infrared communication function. An infrared signal can be output by writing data to bit 0 of register RP. A received infrared signal is latched internally in the CPU by positive edge of the system clock. (System clock goes to HIGH from LOW.) The latched data can be read beginning from bit 1 of register RP by setting bits 6 and 7 to 1.

Note When data is not sent or received, always set the values of register RP to 0x00. This register cannot be written to in DMG mode.



2.6.3.2 Controlling Infrared Communication

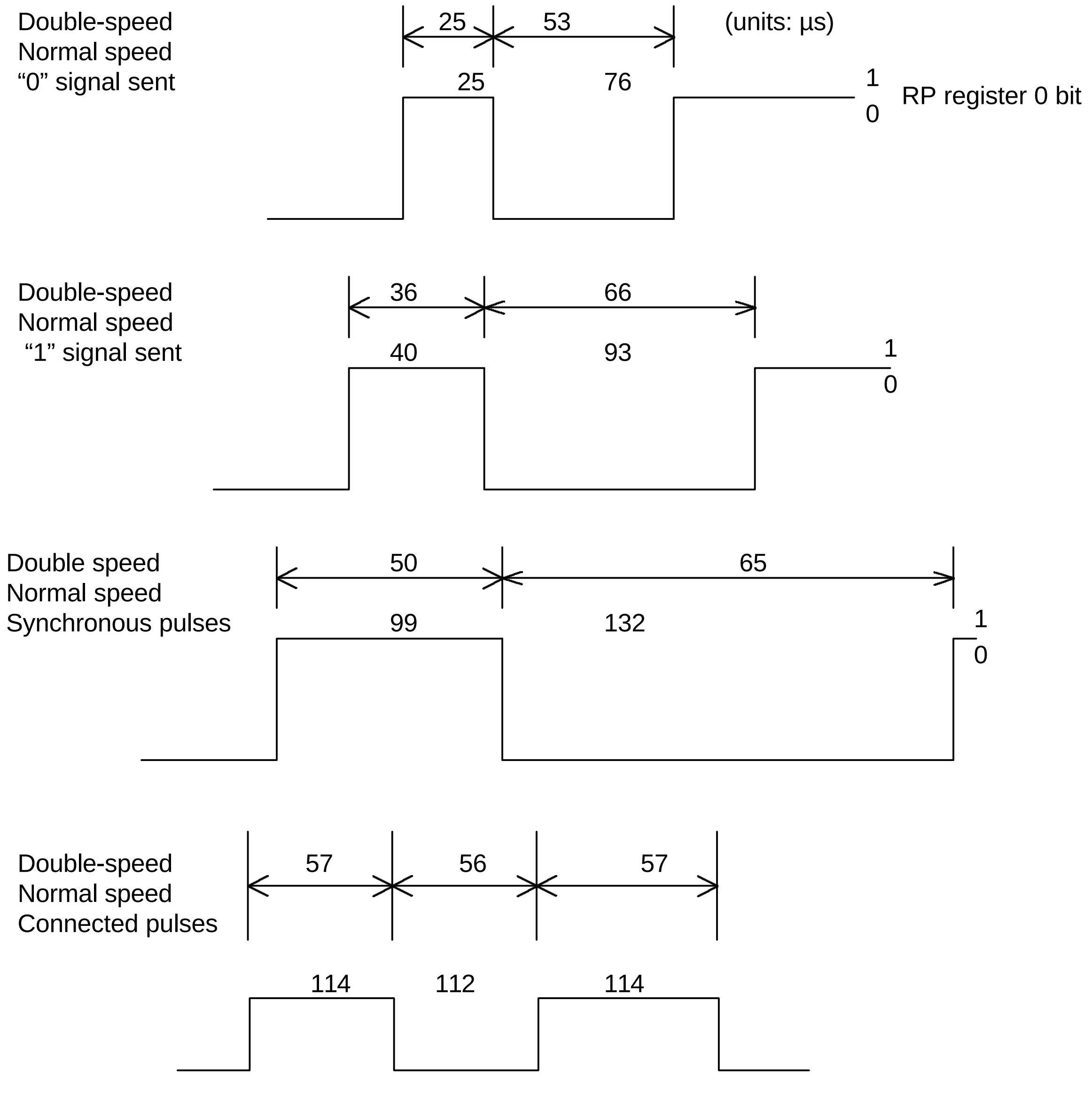
Sender:
Setting bit 0 of the CPU register RP to 1 causes the LED to emit light; setting it to 0 turns off the LED.

Receiver:
If the photo transistor detects infrared light, bit 1 of register RP is set to 0; if no infrared light is detected, this bit is set to 1.

2.6.3.3 Basic Format

When the receiver recognizes the unmodified signal from the sender as a logical value of 1 or 0, the receiver actually cannot distinguish between the continuous transmission of 1s and the absence of received infrared light. The status of the receiver is identical under these conditions. Consequently, to ensure proper data transmission from sender to receiver in Game Boy Color infrared communication, signals are distinguished by the size of the interval between the rising edge of the pulse of one received signal to the rising edge of the subsequent received signal.

The following illustrates signals from a sender.



Scatter in the source oscillation of Game Boy Color produces slight individual differences.

2.6.3.4 Preparing for Data Transmission and Reception

To use infrared communication, data reception must be enabled by setting bits 6 and 7 of Game Boy Color register RP to 1. However, even with both of these bits set to 1, data cannot immediately be received. After setting bits 6 and 7 to 1, at least 50 ms should be allowed to pass before using the infrared port.

2.6.3.5 Transmitted Data

When data is transmitted and received, it is transmitted in packets. Each packet comprises the 4 parts shown below, and each part is sandwiched between synchronous pulses. For more information, see Section 2.6.3.7, *Details of Data Transmission and Reception*.

The data that comprises a packet is transmitted 1 bit at a time beginning from the MSB.

Transmission Packet

Connector	Header	Data	Checksum

Connector:
Signal that implements an infrared communication connection between 2 Game Boy Colors. This is always required in the initial packet. When the receiver receives the connector and recognizes it as a connecting pulse, the receiver returns the same pulse to the sender. The sender then determines whether this signal is a normal connecting pulse. If it is not recognized as a normal pulse, transmission is interrupted at this stage. With continuous communication that is not halted before completion, this part of the packet is unnecessary from the second packet onward.

Header:
Data indicating the type of data being sent and the total number of bytes.

Byte 1: Communication command
0x5A: transmission of raw data
At present, any value other than 0x5A causes an error.
(To be used for by other devices in future)

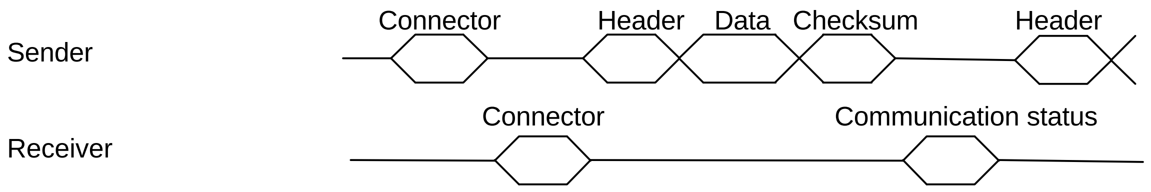
Byte 2: Total number of data in data portion of the packet
0x01-0xFF: Number of data
0x00: Indicates completion of communication to receiver.

Data:
The transmitted data itself. Maximum of 255 bytes.
There are no data if completion of communication is indicated to the receiver.
(The data portion of the packet consists only of a synchronous pulse.)

Checksum:
2 bytes of data consisting of the sum of the header and all data in the data portion of the packet. Following this, the communication status is returned from receiver to sender.

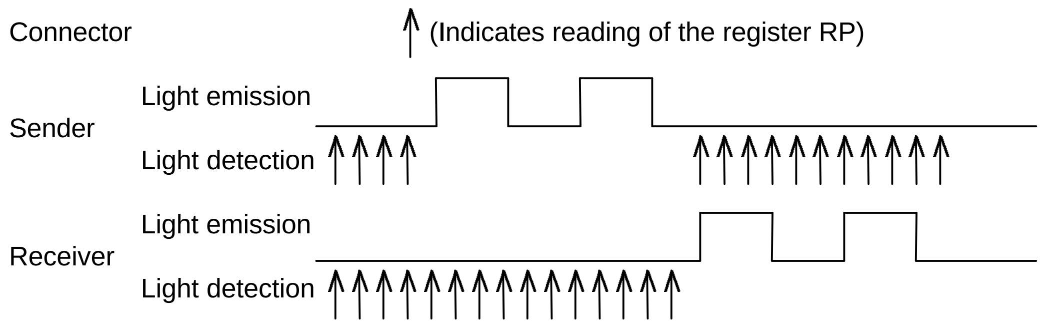
2.6.3.6 Flow of Data Transmission and Reception

When data is transmitted and received, both Game Boy Colors are first placed in receive status. The one with the send indicator is then designated as the sender, and the other one is designated as the receiver. The flow of data transmission is shown below.



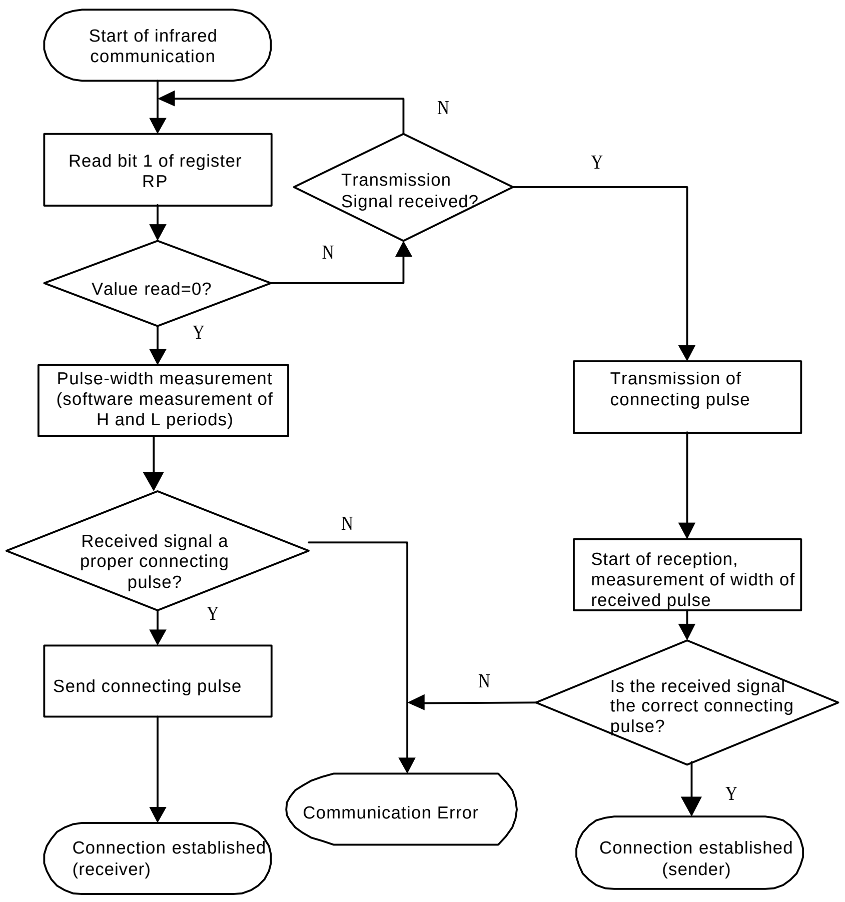
1 Sender transmits connecting pulse.
2 The receiver calculates the width of the received connecting pulse. If the value is correct, the receiver returns the same connecting pulse to the sender.
3 The sender calculates the width of the connecting pulse returned by the receiver. If the value is correct, the sender determines that a connection has been properly established.
4 The header is transmitted.
5 The data is transmitted.
6 The checksum is transmitted.
7 The receiver returns the communication status to the sender.
8 When communication is complete, the header of the subsequently transmitted packet is set to 0x00 + 0x00.

2.6.3.7 Details of Data Transmission and Reception



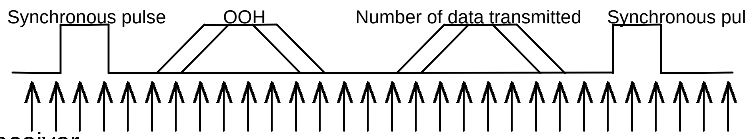
The two Game Boy Colors perform initial data reception, then the one designated as the sender (e.g., by operations such as pressing button A) begins transmission. The connecting portion of the packet is unnecessary from the second packet onward.

The following illustrates the flow for implementing a connection.


Start of infrared communication
Read bit 1 of register RP
Value read=0?
N
Y
Transmission Signal received?
N
Y
Pulse-width measurement (software measurement of H and L periods)
Received signal a proper connecting pulse?
N
Y
Send connecting pulse
Connection established (receiver)
Communication Error
Transmission of connecting pulse
Start of reception, measurement of width of received pulse
Is the received signal the correct connecting pulse?
N
Y
Connection established (sender)

Header

Light emission by sender

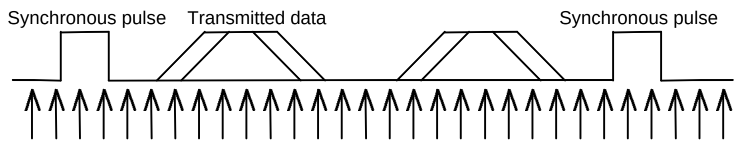


Light detection by receiver

One byte indicating the data type and 1 byte indicating the number of transmitted data are sandwiched between synchronous pulses.

Data

Light emission by sender



Light detection by receiver

Between 1 and 255 bytes of transmitted data are sandwiched between synchronous pulses.

A 2-byte checksum consisting of the sum of the header and transmitted data is sandwiched between synchronous pulses. The receiver uses the checksum to determine whether the transmission was performed properly and notifies the sender of the results of communication status.

The following section describes the details of communication status determination.

2.6.3.8 Communication Status

0x00: Communication OK
0x01: Checksum error

The results of the checksum calculated by the receiver do not agree with the checksum sent by the sender.

In the following cases, the communication status cannot be returned to the sender even if an error is generated during communication (no response from receiver).

- The wrong communication protocol is used.
- Data is transmitted using the wrong pulse width.
- One of them is operating in double-speed mode and the other is operating in normal mode.
- Communication is affected by sunlight or obstruction of the signal light.

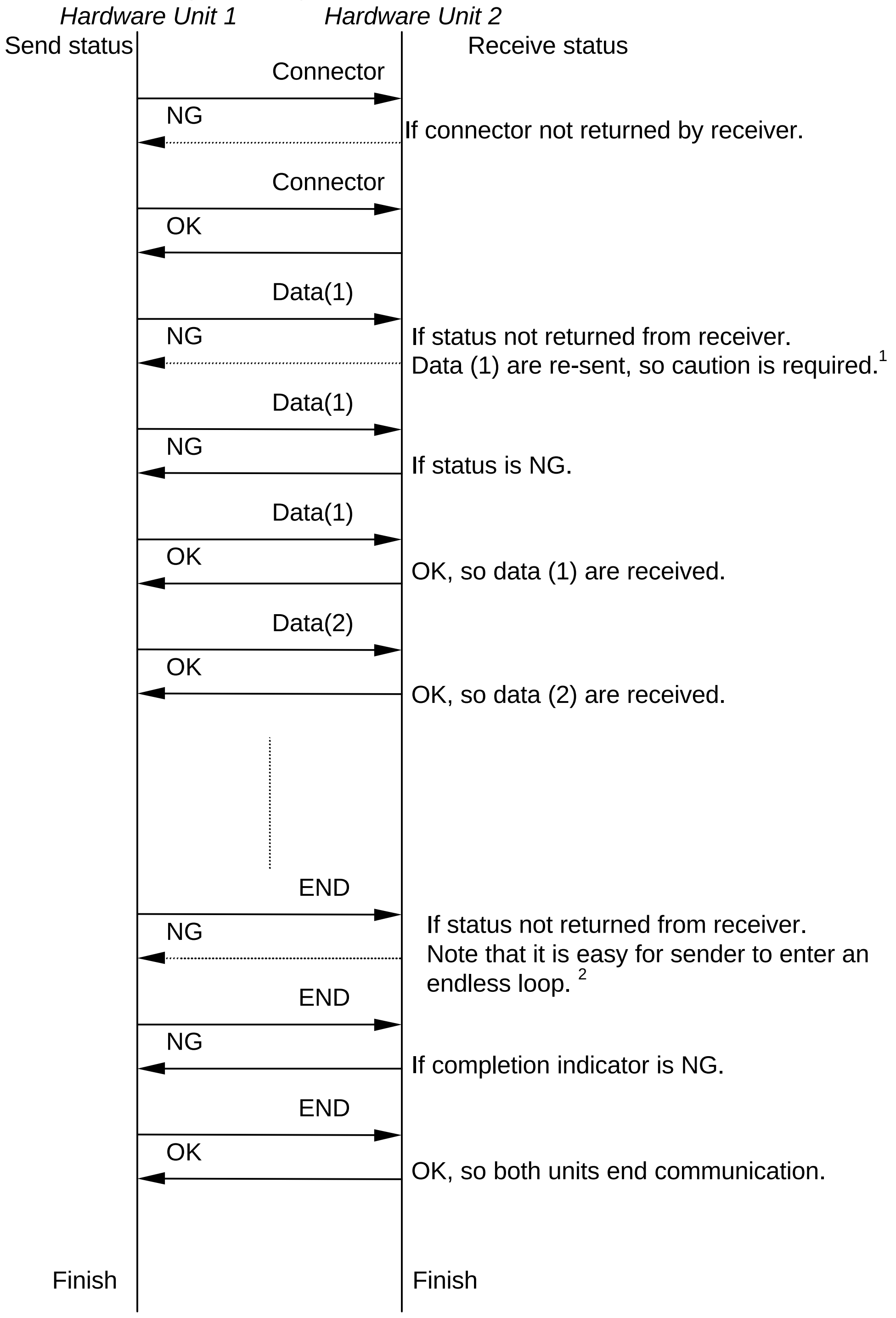
2.6.3.9 Communication Error Processing

If an error described above in Communication Status is generated, the following error codes are returned by subroutine.

Error Code	Error Description
01	Checksum error (same for sender and receiver): The results of the checksum calculated by the receiver and the checksum sent by the sender do not agree.
02	Pulse width error: Generated by the receiver when the width of the pulse of the signal sent by the sender is too wide or narrow. Generated by the sender when the width of the pulse of the signal sent by the receiver is too wide or narrow.
04	Communication error: Communication prevented by other causes. The subroutine provided by Nintendo treats as an error the case when the data value of the second byte of the received header exceeds the number of data items to be received, as determined beforehand by the receiver. The routine also generates an error if the communication command value of byte 1 of the header is not 0x5A.

2.6.3.10 Communication Examples

The following figure shows the flow of processing when errors occur during communication. This should be used as a reference when implementing data communication.



Hardware Unit 1
Hardware Unit 2
Send status
Receive status
Connector
NG
If connector not returned by receiver.
Connector
OK
Data(1)
NG
If status not returned from receiver.
Data (1) are re-sent, so caution is required.[1]
Data(1)
NG
If status is NG.
Data(1)
OK
OK, so data (1) are received.
Data(2)
OK
OK, so data (2) are received.
END
NG
If status not returned from receiver.
Note that it is easy for sender to enter an endless loop. [2]
END
NG
If completion indicator is NG.
END
OK
OK, so both units end communication.
Finish
Finish

1) Data(1) and Data(2) each represent 1 packet for transmission, not including the connector.
2) END signifies the packet used to indicate the completion of transmission (not including the connector).

2.6.3.11 Usage Notes

When programming use of the infrared port, please note the following.

- ♦ When transmitting more than 256 bytes of data, ensure that the receiver keeps track of which packet number is being received. When a communication error (status not returned even though data was received) is generated, the sender will re-send the data, and the receiver may lose track of the packet number (see note 1 of previous section).
- ♦ The sender is prone to entering an endless loop when the packet signifying transmission completion is received. Therefore, the receiver should remain in receive status for approximately 300 µs after returning the status (see note 2 of previous section).
- ♦ Depending on the power reserve of the battery, infrared communication may cause a sudden drop in battery voltage and a complete loss of power.
- ♦ Ensure that the speed of the two communicating Game Boy Colors is the same (both double-speed or both normal speed during communication).
- ♦ Noise can be heard from the speaker and headphones during communication, but this does not indicate a problem with the hardware.
- ♦ Ensure that faulty or uncontrolled operation does not occur when infrared communication signals are input from other game software and devices. Use particular care when using the same subroutine to communicate between various types of games, because fault y or uncontrolled operation is especially likely to occur in such cases. (Before performing data communication, confirm that the other hardware participating in the transmission is using the same game. This can be accomplished by means such as exchanging a unique key code.)

The following are items to note when using an infrared communication subroutine other than that provided by Nintendo.

- ♦ Ensure that error-handling is implemented to prevent the program from entering an endless loop when communication is interrupted by sunlight or obstruction of the signal light.
- ♦ To reduce power consumption, use a maximum infrared LED emission pulse duration of 150 µs and a duty ratio of approximately 1/2.
- ♦ Do not leave the infrared LED or photo transistor ON when not using infrared communication.

2.6.3.12 Specifications

1) Communication Speed
   Normal-speed mode: approximately 7.5 Kbps
   Double-speed mode: approximately 10.5 Kbps
2) Communication distances: Minimum, 10 cm, Typical, 15 cm
3) Recommended directional angle: approximately ± 15°