Title: GB Printer interface specification
Author: anonymous
First version: Sep. 8, 2001
Last updage: Dec. 1, 2001

1. INTRODUCTION

This document describes *my* understanding of the communication protocol and data format used by GB Printer, which is an optional peripheral of Nintendo's Gameboy portable consoles.

The information provided here is based on various inputs from Internet as well as my own experiments. See the references given below for the sources.

This document does not discuss the hardware details on GB-GBP communication, i.e., the document doesn't include information like: bit-by-bit transmission mechanism, cable and/or connector specifications, and electrical characteristics of signals. Consult somewhere else for those things.

2. COMMUNICATION FRAMEWORK

When a GB communicate with a GB Printer, the GB always becomes a master (an entity drives a clock signal) and the printer becomes a slave (an entity receives a clock signal.) There are no roll (master or slave) arbitration phase in communication.

GB communication cable and its low-level protocol provide byte-oriented full-duplex bi-directional communication channel as always. However, as you will see in the following discussion, the full-duplexity is not used in GB Printer. When a GB is sending some information to GB Printer, bytes sent back from GB Printer to GB is just ignored. Conversely, when a GB is receiving some information from GB Printer, bytes sent from GB to GB Printer is just ignored.

The communication consists of several command packets (sent from GB to GB Printer) and acknowledgements (sent from Printer to GB.) The communication is initiated by GB, i.e., GB may send a command packet anytime. The printer must and acknowledge after a command packet has been received.

A command packet has variable length. It consists of eight bytes or more. (Up to probably 648 bytes.) An acknowledgement always consists of two bytes.

When GB is sending a command packet, Printer sends zeros continuously, although it seems that these zeros are just ignored and may be other value. After a command packet has been sent, GB sends two zero bytes to generate clock pulses to receive an acknowledgement from the printer.

3. COMMAND PACKET FORMAT

A command packet consists of the following four parts: a Synchronization mark, a Header, a Body, and a Checksum, in this order. Note that, in this document, we consider that the two zeros sent by GB to receive acknowledgement from the printer are *not* part of the command packet.

The four parts in a command packet are as follows:

A Synchronization mark consists of two fixed bytes: 0x88 0x33. This sequence indicates the beginning of a command packet.

A Header consists of four bytes. The first byte in the Header is a command code, and the second is a compression indicator. Third and fourth bytes forms a 16 bit integer (in LSB-first byte order, i.e., the third byte is the lower 8 bits and the fourth the higher)

representing the length of the Body of the command packet in bytes. This value may be zero to indicate an empty Body.

A Body consists of zero or more bytes. The length field in the Header indicates the length of the Body. The meaning of each byte in Body varies depending on command codes. The length field in Header may be set to zero. In that case, the Body is empty, and the Checksum immediately follows Header.

A Checksum consists of two bytes. They form a 16 bit integer (in LSB-first byte order again) representing a checksum of the command packet.

See the following chart for a brief idea of the command packet format.

```
        +--------+ Synchronization mark:
        |10001000|      fixed value (0x88)
        |00110011|      fixed value (0x33)
        +--------+ Header:
        |0000xxxx|      command code
        |0000000x|      compression indicator
        |xxxxxxxx|      length of Body, lower byte
        |xxxxxxxx|      length of Body, higher byte
        +--------+ Body:
        |        |
        :    :   :      0 or more bytes of various data
        |        |
        +--------+ Checksum:
        |xxxxxxxx|      lower byte
        |xxxxxxxx|      higher byte
        +--------+
```

3.1. COMMAND CODES

A command code identifies the purpose of the command packet. Currently, there are four known command codes: Initialize (0x01), Data (0x04), Print (0x02), and Inquiry (0x0F).

When GB prints, it sends several commands in sequence: One Initialize command, one or more (usually nine) printing Data commands, one empty Data command, and one Print command, in this order. At any point in the sequence, GB may optionally insert some Inquiry commands.

3.2. INITIALIZE COMMAND

A command packet whose command code is set to 0x01 is an Initialize command. The length field in Header of an Initialize command is always zero (an empty Body.) The compression indicator field in Header is always zero, too.

GB sends one Initialize command when it starts printing a page. If it attempts to print several pages in sequence, one Initialize command is sent for each page.

The *real* purpose of this command is unknown.

3.3. DATA COMMAND

A command packet whose command code is set to 0x04 is a Data command.

Currently, there are three different known types of Data commands: plain printing Data command, compressed printing Data command, and empty Data command.

A plain printing Data command is used to transmit (a part of) printing image from GB to Printer. The compression indicator in Header is set to zero (0x00.) The length field in Header is set to 0x280, or 640 in decimal. Since 16 bit values are transmitted in LSB-first byte order, it is actually transmitted as 0x80 0x02. The Body in plain printing

Data command represents a band (160x16 dots or 20x2 tiles) of printing image in plain (uncompressed) format.

A compressed printing Data command is used to transmit (a part of) printing image from GB to Printer. The Body of a compressed printing Data command is compressed using a sort of run-length compression method to reduce the transmission overhead. The compression indicator in Header is set to one (0x01.) The length field in Header varies, indicating the number of bytes of printing image *after* compression. The Body in compressed printing Data command represents a band (160x16 dots or 20x2 tiles) of printing image in compressed format.

An empty Data command indicates the end of a page. The compression indicator in Header is set to zero (0x00,) and the length field too (0x00 0x00.) One empty Data command is sent after the last printing Data command for a page and before the Print command for the page. The purpose of this command is unknown.

3.4. PRINT COMMAND

A command packet whose command code is set to 0x02 is a Print command. The compression indicator in Header of a Print command is always zero (0x00.) The length field in Header of a Print command is always four (transmitted as 0x04 0x00, since it is LSB-first 16 bit value,) i.e., Body in a Print command is always four bytes in length.

A Print command is issued after sufficient number of printing Data commands have been issued. When the Print command is received, the GB Printer physically starts printing. In other words, printing image data sent between the last Initialize command and this Print command collectively form a page.

The Body of a Print command consists of four bytes: The first byte is always set to 0x01, and its purpose is unknown. The second byte represents margins before and after the page. The third byte defines the palette to be used to render the page. The fourth byte represents the dense of ink for printing (or the heat at the printing head since real GB Printer is a thermal printer.)

The margins byte (the second byte in Body) is divided into two four-bit values. The upper four-bits represents the margin before the page, and the lower four-bits represents the margin after the page. The four bit values control margins as follows: 0x0 is for no margin, 0x1 for minimum margin, and 0xF for maximum margin. Any value between 0x1 and 0xF is for some margin between minimum and maximum, apparently proportional to the specified value.

Unlike other commands, a Print command requires long time to complete, since the command requires some mechanical actions to complete. GB games that print usually issue Inquiry commands periodically after a Print command to wait for the command to complete.

3.5. INQUIRY COMMAND

A command packet whose command code is set to 0x0F is an Inquiry command. The compression indicator of the Inquiry command is set to zero (0x00.) The length field is always set to zero (0x00 0x00,) to indicate the Body in the command is empty.

The purpose of Inquiry command is to make GB Printer to notify its status to the GB. So, this command is suitable to be used after a Print command or before an Initialize command. However, unlike other commands, Inquiry command may be issued at any time. The GB Printer is expected to respond to an Inquiry command always.

3.6. CHECKSUM

A Checksum is a two-byte (16 bit) value to verify a command packet is

received properly. It is always appended at the end of a command packet.

The Checksum is a 16 bit arithmetic sum over bytes in the command packet, excluding the Synchronization marker (i.e., 0x83 and 0x33 at the beginning of the packet) and the Checksum, regarding each byte as an eight-bit unsigned integer.

4. ACKNOWLEDGEMENT

An acknowledgement is a two-byte data sent from GB Printer to GB after a command packet to notify the status of the Printer. The first byte is the acknowledgement code and the second is the status code.

4.1. ACKNOWLEDGEment CODE

An acknowledgement code is set (by GB Printer) to either 0x80 or 0x81. The difference of those two values is unsure at this moment. However, any other values are unaccepted and should be avoided.

When writing a GB game, it is safe to ignore these two values identical (i.e., masking with 0xFE.)

When writing a GB Printer alternative (e.g., emulator,) it is safe to return 0x81 always.

4.2. STATUS CODE

A status code is a bitmap to indicate various Printer statuses. It has bit-by-bit meanings. Zero-valued bit represents normal status. One bit represents some unusual status as follows:

When the highest bit is set, the status is called "error #1," meaning Low Batteries.

When the second highest bit is set, the status is called "error #4," meaning the heat level at the printing head is unusual (too hot or too cold.)

When the third highest bit is set, the status is called "error #3," meaning the printer paper jam.

When the second lowest bit (i.e., sixth highest bit) is set, the status is called "busy," meaning that the mechanics of the printer is working (probably as the result of the last Print command.) This status is not an error, unless a printer keeps reporting this status for a long periods.

The meanings of other bits are unknown. (Except that, in my experience, other bits look never be set.) Note that the status called "error #2" is detected by GB games as a result of a "timeout," as which GB detect the case that an invalid acknowledgement code (any value other than 0x80 or 0x81) is received after a command packet.

5. IMAGE DATA FORMAT

GB Printer uses a dedicated roll paper for print out. The paper has a fixed width and very long (indefinite) length. So, the width of a printing image is fixed to 160 dots (pixels.) However, since roll paper may be cut at any point, the height can be very flexible.

Printing image data contained in a Data command are decoded into 40 tiles of 8x8 dots each. A fragment of image of size 160x16 dots, which is filled with exactly 40 tiles, is known as a band. A sequence of one to nine bands forms a page. And, lastly, one or more pages contiguously printed on a roll paper form an image.

5.1. BAND

An (either plain or compressed) printing Data command always contains data representing an image of size 160x16 dots. This image fragment is known as a band.

Since a band consists of 40 tiles (a tile represents 8x8 dots of image), and a tile occupies 16 bytes, the Body of a plain printing Data command is always 640 bytes in length.

The length of Body of a compressed printing Data command varies, but it is always decompressed into 640 bytes, forming a band.

The band is the minimum image fragment that can be passed from GB to GB Printer.

5.2. PAGE AND IMAGE

The GB Printer hardware has a limit of 144 dots (pixels) in length (height) in its internal buffer size. In other words, GB Printer can keep unprinted image data consisting of up to nine bands before a Print command is issued. So, a Print command must be issued for every nine (or less) print Data commands.

A page is defined as a part of image whose width is 160 dots and whose height is less than 144 dots (and a multiple of 16 dots.) Between an Initialize command and a Print command, GB sends one to nine printing Data commands to pass image data forming a page.

When a GB game prints an image larger than a page, it breaks the image into several pages. In this case, the before-margin of each Print command except for the last page and the after-margin of each Print command except for the first page are all set to zeroes. This way, those pages forming one image are printed contiguously.

5.3. TILE FORMAT

A band consists of 40 tiles. A tile is an image consisting of 8x8 dots. Color of each dot is specified by a two-bit value, so a dot can be a color out of four. Actually, GB Printer prints only greyscale (monochrome) image, so possible colors are: white, light grey, dark grey, and black.

A tile is represented by 16 bytes of data. The format of a tile is exactly same as that of GB display tile pattern, such as background tile, window tile, or sprite tile.

5.4. REPRESENTATION IN PLAIN PRINTING DATA COMMAND

In a plain printing Data command, the Body of the command packet contains 640 bytes for a band (40 tiles.) The format is simple; 640 bytes of data are divided into 40 data blocks of 16 bytes. Each 16-byte data block represents a tile in standard GB tile pattern format. 40 tiles are laid out on paper as follows: the first 20 tiles are printed from left to right on the upper half of the band, and the next 20 tiles are on the lower half from left to right.

5.5. REPRESENTATION IN COMPRESSED PRINTING DATA COMMAND

In a compressed printing Data command, the Body of the command packet contains 40 tile pattern data in run-length compressed form.

The idea of run-length compression is as follows: First, an image data for a band is prepared as its uncompressed form, i.e., as a simple sequence of 40 tile pattern data. Then, the data (40 tiles or 640 bytes) is scanned from top to bottom. If there are two or more bytes of the same value occur contiguously, Replace the subsequence with a [length, byte] pair, where the length is the number of bytes in the subsequence and the byte is the byte value that occurred contiguously. The resulting pair is called a compressed run. After all of such subsequences are replaced, each of the remaining uncompressible

subsequence is prefixed by the length in bytes of that uncompressible subsequence. The uncompressible subsequence prefixed by a length is called an uncompressed run.

For example, if the first 16 bytes of the 640 bytes of printing data was:

    00 00 00 00 FF 00 FF 00 00 FF 00 FF FF FF FF FF

The corresponding run-length compression will be:

    [4,00] (7) FF 00 FF 00 00 FF 00 [5,FF]

Where [4,00] and [5,FF] are compressed runs, and (7) FF 00 FF 00 00 FF 00 is an uncompressed run.

In the Body of a compressed printing Data command, each run is encoded into a sequence of several bytes. The MSB of the first byte in the run identifies the type of the run: If MSB is set to one, the run led by the byte is compressed, otherwise the run is uncompressed.

A compressed run is always encoded as two bytes. The MSB of the first byte is set to one to indicate that the run is compressed. Lower seven bits in the first byte indicates the number of expanded bytes corresponding to this run; the value represented as the seven bit unsigned integer plus two is the number of contiguous bytes represented by this run. The second byte is the byte to be copied.

An uncompressed run is encoded as two or more bytes. The MSB of the first byte in the run is set to zero to indicate that the run is uncompressed. Lower seven bits in the first byte indicates the number of bytes following the first byte in this run; the value represented as the seven bit unsigned integer plus one is the number of bytes following the first byte.

The 16 bytes in the above example will be encoded as:

    82 00 06 FF 00 FF 00 00 FF 00 83 FF

5.6. NOTES ABOUT RUN-LENGTH COMPRESSION

There are some additional notes on the details of run-length compression.

There may be some upper bound on the maximum length of a compressed run. Experiences showed that the maximum length of the expanded bytes from a compressed run may be bounded by 32. This means the maximum value of the first byte for a compressed run is 0x9E. For an uncompressed run, there is no such limit.

A run may span over different tile patterns. A run, regardless of compression, may contain bytes from two (or more) different tiles. In other words, bytes forming one tile pattern may be split into several runs.

5.7. PALETTE MAPPING

A tile pattern represents an 8x8 array of two bit integer values. Each two-bit value is an index to palette table. In GB LCD, the palette tables are located on several LCD controller registers, e.g., on register at 0xFF47. In GB Printer, the palette table is passed from GB as a part of a Print command. The palette table determines brightness (or blackness, since GB Printer prints greyscale graphics on white paper) of each dot in tile pattern.

The palette table is one-byte data, consisting of four two-bit values. The highest two bits in a palette byte represent the brightness of dots with index 0. The next two bits represent those of index 1, the next, index 2, and the lowest two bits, index 3.

A two-bit entry in palette table of value 0 represents white, i.e.,
nothing is printed at the corresponding point on paper. That of value
3 represents black, i.e., darkest dot is printed. Those of values 1 or
2 represent greys, where 1 is lighter and 2 is darker.

On GB Printer, there is no coloring information (something used by
Super GB or GB Color) passed.

6. REFERENCES

[1] kOOPa, Everything You Always Wanted To Know About GAMEBOY, Mar. 1998,
    http://www.devrs.com/gb/files/gbspec.txt

[2] Martin Eyre, Gameboy Printer Format, Aug. 1998,
    Distributed with "Gameboy Printer Emulator, Version 1.3, Sep. 1998,"
    on http://skyscraper.fortunecity.com/macro/730/

[3] Jeff Frohwein, GameBoy Dev'rs, (updated frequently),
    http://www.devrs.com/gb/
 LocalWords: xxx xxxxxxxx FF FF FF FF FF FF FF FF MSB