# Game Boy Camera Technical Information v1.0.1

by Antonio Niño Díaz (AntonioND)

antonio_nd@outlook.com

03/2015

# 1 Introduction

Some years ago (about 2010) I tried to emulate the Game Boy Camera by using its ROM and the sensor datasheet (which is really bad). There was a lot to guess so I couldn't emulate it completely. Since there is no documentation about the Game Boy Camera I decided it was time to document it, but I really couldn't do much without doing tests in the cartridge itself.

This time I've used Arduino to send commands to the cartridge and to read the sensor pins to know how they are translated. This means that I can know how the controller works so I only need to know how the sensor works. I've used the datasheets from sensors M64283FP and M64285FP (the Game Boy Camera has a M64282FP) to understand the things I couldn't before. You should have received this document alongside M64282FP, M64283FP, M64285FP and M64285K datasheets.

I've tried to explain all of the inner workings, enough to interface the cartridge using custom hardware or software and, of course, enough to emulate it correctly!

# 2 The Game Boy Camera Cartridge

The Game Boy Camera cartridge contains 4 ICs: the usual ROM and RAM ICs, a big controller IC (like a MBC) and a sensor (M64282FP "retina" chip).

The main board contains all ICs except from the sensor. From GBDev wiki:

| Component# | Part#/inscription | Description |
|---|---|---|
| U1 | MAC-GBD<br>Nintendo<br>9807 SA | I/O, memory control. |
| U2 | GBD-PCAX-0 F<br>M538011-E - 08<br>8145507 | 1MB ROM |
| U3 | 52CV1000SF85LL<br>SHARP JAPAN<br>9805 5 0A | 128KB RAM |

The U1 is the only one connected to the GB cartridge pins (besides some of the address pins of the ROM IC). The U2 and U3 (ROM and RAM) are connected to U1. The M64282FP "retina" chip is in a separate PCB, and is connected to the U1.

The M64282FP handles most of the configuration of the capturing process. The U1 transforms the commands from the Game Boy CPU into the correct signals needed for the M64282FP. The detailed timings are described below.

It is a good idea to have the datasheet of the M64282FP, but it is very poorly explained, so this document will try to explain everything about it (except from limits like voltage or signal timings). There are datasheets of similar sensors (M64283FP and M64285FP) that can be very useful to understand some things about the sensor of the GB Camera.

## 2.1 Game Boy Camera MBC

The Game Boy Camera controller works pretty much the same as a MBC3.

**0000-3FFF - ROM Bank 00 (Read Only)**

First 16 KB of the ROM.

**4000-7FFF - ROM Bank 01-3F (Read Only)**

This area may contain any ROM bank (0 included). The initial mapped bank is 01.

**A000-BFFF - RAM Bank 00-0F (Read/Write)**

**A000-BFFF - CAM Registers (Read/Write)**

Depending on the current RAM Bank Number, this memory space is used to access the cartridge RAM or the CAM registers. RAM can only be read if the capture unit is not working, it returns 00h otherwise.

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

A value of 0Ah will enable writing to RAM, 00h will disable it. Reading from RAM or registers is always enabled. Writing to registers is always enabled. Disabled on reset.

**2000-3FFF - ROM Bank Number (Write Only)**

Writing a value of 00-3Fh selects the corresponding ROM Bank for area 4000-7FFF.

**4000-5FFF - RAM Bank Number/CAM Registers Select (Write Only)**

Writing a value in range for 00h-0Fh maps the corresponding external RAM Bank to memory at A000-BFFF. Writing any value with bit 5 set to '1' will select CAM registers. Usually bank 10h is used to select the registers. All registers are mirrored every 80h bytes. RAM bank 0 selected on reset.

IMPORTANT NOTE: Unlike most games, the GB Camera RAM can only be written when PHI pin = '1'. It's an enable signal for the RAM chip. Most cartridge readers and writers can't handle PHI pin so they can't restore a saved backup. It isn't needed to change ROM banks.

## 2.2 Game Boy Camera I/O Registers

The Game Boy Camera I/O registers are mapped to all banks with bit 4 set to '1'. The GB Camera ROM usually changes to bank 16 to use the registers.

There are 3 groups of registers:

- The first group is composed by the trigger register A000. This register starts the capture process and returns the current status (working/capture finished).
- The second group is composed by registers A001-A005, used to configure most parameters of the M64282FP sensor.

- The third group is composed by 48 registers that form a 4×4 matrix. Each element of the matrix is formed by 3 bytes. This matrix is used by the controller for contrast and dithering.

All registers are write-only, except the register A000. The others return 00h when read. The initial values of all registers on reset is 00h.

### 2.2.1 Register A000

The lower 3 bits of this register can be read and write. The other bits return '0'. Writing any value with bit 0 set to '1' will start the capturing process. Any write with bit 0 set to '0' is a normal write and won't trigger the capture. The value of bits 1 and 2 affects the value written to registers 4, 5 and 6 of the M64282FP, which are used in 1-D filtering mode (effects described in following chapters).

Bit 0 of this register is also used to verify if the capturing process is finished. It returns '1' when the hardware is working and '0' if the capturing process is over.

When the capture process is active all RAM banks will return 00h when read (and writes are ignored), but the register A000 can still be read to know when the transfer is finished.

The capturing process can be stopped by writing a '0' to bit 0. When a '1' is written again it will continue the previous capture process with the old capture parameters, even if the registers are changed in between. If the process is stopped RAM can be read again.

### 2.2.2 Register A001

This register is mapped to register 1 of M64282FP. It controls the output gain and the edge operation mode.

### 2.2.3 Registers A002, A003

This registers are mapped to registers 2 and 3 of M64282FP. They control the exposure time. Register 2 is the MSB, register 3 is the LSB.

```
u16 exposure_steps = [A003] | ([A002]<<8);
```

### 2.2.4 Register A004

This register is mapped to register 7 of M64282FP. It sets the output voltage reference, the edge enhancement ratio and it can invert the image.

### 2.2.5 Register A005

This register is mapped to register 0 of M64282FP. It sets the output reference voltage and enables the zero point calibration.

### 2.2.6 Registers A006-A035

Those registers form a 4×4 matrix with 3 bytes per element. They handle dithering and contrast, and they are sorted by rows:

| | X | | | |
|---|---|---|---|---|
| Y | 00 | 10 | 20 | 30 |
| | 01 | 11 | 21 | 31 |
| | 02 | 12 | 22 | 32 |
| | 03 | 13 | 23 | 33 |

| **A006** | **A007** | **A008** | **A009** | **A00A** | **A00B** | **A00C** | **A00D** | **A00E** | **A00F** | **A010** | **A011** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D00L | D00M | D00H | D10L | D10M | D10H | D20L | D20M | D20H | D30L | D30M | D30H |
| **A012** | **A013** | **A014** | **A015** | **A016** | **A017** | **A018** | **A019** | **A01A** | **A01B** | **A01C** | **A01D** |
| D01L | D01M | D01H | D11L | D11M | D11H | D21L | D21M | D21H | D31L | D31M | D31H |
| **A01E** | **A01F** | **A020** | **A021** | **A022** | **A023** | **A024** | **A025** | **A026** | **A027** | **A028** | **A029** |
| D02L | D02M | D02H | D12L | D12M | D12H | D22L | D22M | D22H | D32L | D32M | D32H |
| **A02A** | **A02B** | **A02C** | **A02D** | **A02E** | **A02F** | **A030** | **A031** | **A032** | **A033** | **A034** | **A035** |
| D03L | D03M | D03H | D13L | D13M | D13H | D23L | D23M | D23H | D33L | D33M | D33H |

This values are used for the contrast and dithering effects in the following way for each pixel at position (X,Y):

- The sensor outputs an analog value. This value is compared to the corresponding group of 3 values of the matrix probably using 3 DACs and 3 comparators (clock timings doesn't affect the result when converting from analog to digital, most ADCs would be affected).
- This value is compared to the corresponding group of 3 values of the matrix. To know what group of 3 values to use do (X mod 4) and (Y mod 4).
- The controller performs the following operations:

```
if ( sensor_value < DxyL ) gb_shade = black;
else if ( sensor_value < DxyM ) gb_shade = dark_gray;
else if ( sensor_value < DxyH ) gb_shade = light_gray;
else gb_shade = white;
```

This means that the minimum value to show a different color than white is 01h. A value of 00h in the 3 values will always show a white screen. A value of FFh in the 3 bytes will always allow some white pixels in a black background.

If two values are the same, the second comparison will be ignored. If the values are reversed (the higher bytes have lower values) that values are ignored. The previous comparisons explain that kind of behaviors.

To disable dithering all groups of 3 values must be the same. Dithering is obtained by doing small variations in the values of each one of the 4×4 groups. Contrast is handled by modifying the distance between values. Max contrast is reached when the three values of each group are the same. In that case the resulting image has only black and white pixels.

The digital-to-analog conversion is not linear in the sense that a signal of 0V doesn't correspond to a digital value of 00h, it's affine: A value of 00h is around 0.8V and a value of FFh is around

4.1V. Note that the sensor has a 5.0V supply. That means that the value range is approximately 3.3V, but it can't be reached by the sensor output. The sensor can only output from approximately 0.9V to 4.0V (by modifying Vref and other configuration values). The previous limits have been determined by extrapolating, since the values are completely linear (if the offset at 00h is ignored).

According to the sensor datasheet the output voltage range is 2.0V peak to peak. The actual range seems to be up to 3.15V depending on the register configurations. That's the range of values that the controller has to convert into 4 Game Boy gray shades.

The following table shows sample values for the matrix registers. The values are used with dithering enabled, to disable it just repeat the upper left values in all the matrix.

| Matrix set 1 (high light) | | | | Matrix set 2 (low light) | | | |
|---|---|---|---|---|---|---|---|
| Lowest contrast | | | | | | | |
| 80 | 8F | D0 | 8B BF E0 | 82 | 9B | D4 | 8E CB E4 |
| 87 | AF | DB | 83 9F D5 | 8A | BB | DF | 86 AB D9 |
| 81 | 97 | D2 | 8D C7 E3 | 80 | 93 | D1 | 8C C3 E1 |
| 89 | B7 | DD | 85 A7 D8 | 88 | B3 | DC | 84 A3 D6 |
| 80 | 94 | DC | 8F CA F6 | 83 | A1 | E2 | 92 D7 FC |
| 8A | B8 | ED | 85 A6 E4 | 8D | C5 | F4 | 88 B3 EB |
| 82 | 9D | E0 | 91 D3 FA | 81 | 98 | DE | 90 CE F8 |
| 8C | C1 | F1 | 87 AF E9 | 8B | BC | EF | 86 AA E6 |
| 82 | 90 | C8 | 8C BA DC | 84 | 9A | CD | 8F C4 E1 |
| 89 | AC | D5 | 85 9E CE | 8B | B6 | DA | 88 A8 D3 |
| 83 | 97 | CB | 8E C1 DF | 82 | 93 | C9 | 8D BD DD |
| 8A | B3 | D8 | 87 A5 D2 | 89 | AF | D7 | 86 A1 D0 |
| 82 | 95 | D2 | 90 C2 F3 | 85 | A0 | DA | 93 CE FC |
| 8B | B3 | E8 | 86 A4 DD | 8F | BE | F0 | 8A AF E5 |
| 84 | 9C | D7 | 92 CA F9 | 83 | 98 | D4 | 91 C6 F6 |
| 8D | BB | EE | 89 AB E2 | 8C | B7 | EB | 87 A8 E0 |
| 84 | 90 | C0 | 8D B4 D8 | 86 | 99 | C6 | 8F BD DE |
| 8A | A8 | D0 | 87 9C C8 | 8C | B1 | D6 | 89 A5 CE |
| 85 | 96 | C4 | 8E BA DC | 84 | 93 | C2 | 8D B7 DA |
| 8B | AE | D4 | 88 A2 CC | 8A | AB | D2 | 87 9F CA |
| 84 | 96 | CA | 91 BD F1 | 87 | 9F | D3 | 94 C6 FB |
| 8D | B0 | E4 | 88 A3 D7 | 90 | B9 | EE | 8B AC E1 |
| 86 | 9C | D0 | 93 C3 F8 | 85 | 99 | CD | 92 C0 F5 |
| 8F | B6 | EB | 8A A9 DD | 8E | B3 | E7 | 89 A6 DA |
| 85 | 91 | B8 | 8E AE D3 | 87 | 98 | BE | 90 B5 DA |
| 8B | A4 | CA | 88 9A C1 | 8D | AB | D1 | 8A A2 C8 |
| 86 | 95 | BC | 8F B3 D8 | 85 | 93 | BA | 8E B0 D6 |
| 8C | A9 | CF | 89 9F C5 | 8B | A6 | CC | 88 9D C3 |
| 86 | 96 | C4 | 92 B8 F0 | 89 | 9E | CF | 95 C1 FB |
| 8E | AD | E1 | 8A A1 D2 | 91 | B5 | EC | 8D AA DD |
| 88 | 9B | CB | 94 BE F7 | 87 | 98 | C7 | 93 BB F3 |
| 90 | B2 | E8 | 8C A7 DA | 8F | AF | E5 | 8B A4 D6 |
| 86 | 91 | B1 | 8E A9 D0 | 88 | 97 | B8 | 90 AF D8 |
| 8B | A1 | C6 | 88 99 BB | 8D | A7 | CD | 8A 9F C3 |
| 87 | 95 | B6 | 8F AD D5 | 86 | 93 | B3 | 8E AB D3 |
| 8C | A5 | CB | 8A 9D C0 | 8C | A3 | C8 | 89 9B BE |
| 88 | 97 | BE | 93 B4 EE | 8A | 9E | CA | 96 BB FA |
| 8F | AA | DE | 8B A0 CE | 92 | B1 | EA | 8E A8 DA |
| 89 | 9B | C6 | 95 B9 F6 | 88 | 99 | C2 | 94 B6 F2 |
| 91 | AF | E6 | 8D A5 D6 | 90 | AC | E2 | 8C A3 D2 |

| | | | |
|---|---|---|---|
| 87 | 92 | AA | |
| 8C | 9E | C1 | |
| 88 | 95 | AF | |
| 8D | A1 | C6 | |

| | | | |
|---|---|---|---|
| 8F | A4 | CC | |
| 89 | 98 | B5 | |
| 90 | A7 | D2 | |
| 8B | 9B | BB | |

| | | | |
|---|---|---|---|
| 89 | 96 | B2 | |
| 8E | A2 | C9 | |
| 87 | 93 | AC | |
| 8D | 9F | C3 | |

| | | | |
|---|---|---|---|
| 91 | A8 | D5 | |
| 8B | 9C | BE | |
| 8F | A5 | CF | |
| 8A | 99 | B8 | |

| | | | |
|---|---|---|---|
| 8A | 97 | B8 | |
| 90 | A7 | DB | |
| 8B | 9B | C0 | |
| 92 | AB | E4 | |

| | | | |
|---|---|---|---|
| 93 | AF | ED | |
| 8D | 9F | C9 | |
| 95 | B3 | F6 | |
| 8E | A3 | D2 | |

| | | | |
|---|---|---|---|
| 8C | 9D | C5 | |
| 92 | AD | E8 | |
| 8A | 99 | BC | |
| 91 | A9 | DF | |

| | | | |
|---|---|---|---|
| 96 | B5 | FA | |
| 8F | A5 | D7 | |
| 94 | B1 | F1 | |
| 8E | A1 | CE | |

| | | | |
|---|---|---|---|
| 88 | 92 | A5 | |
| 8D | 9B | BD | |
| 89 | 94 | AB | |
| 8E | 9D | C3 | |

| | | | |
|---|---|---|---|
| 8F | A0 | C9 | |
| 8A | 96 | B1 | |
| 90 | A2 | CF | |
| 8B | 99 | B7 | |

| | | | |
|---|---|---|---|
| 89 | 95 | AE | |
| 8E | 9F | C6 | |
| 88 | 93 | A8 | |
| 8D | 9C | C0 | |

| | | | |
|---|---|---|---|
| 91 | A3 | D2 | |
| 8C | 9A | BA | |
| 90 | A1 | CC | |
| 8B | 97 | B4 | |

| | | | |
|---|---|---|---|
| 8B | 98 | B2 | |
| 91 | A5 | D3 | |
| 8C | 9B | BA | |
| 93 | A8 | DB | |

| | | | |
|---|---|---|---|
| 94 | AB | E4 | |
| 8E | 9E | C2 | |
| 96 | AE | EC | |
| 8F | A1 | CB | |

| | | | |
|---|---|---|---|
| 8D | 9C | BE | |
| 93 | A9 | E0 | |
| 8B | 99 | B6 | |
| 92 | A6 | D7 | |

| | | | |
|---|---|---|---|
| 97 | B0 | F0 | |
| 90 | A3 | CF | |
| 95 | AD | E8 | |
| 8F | A0 | C6 | |

Default contrast

| | | | |
|---|---|---|---|
| 89 | 92 | A2 | |
| 8D | 9A | BA | |
| 8A | 94 | A8 | |
| 8E | 9C | C0 | |

| | | | |
|---|---|---|---|
| 8F | 9E | C6 | |
| 8B | 96 | AE | |
| 90 | A0 | CC | |
| 8C | 98 | B4 | |

| | | | |
|---|---|---|---|
| 8A | 95 | AB | |
| 8F | 9D | C3 | |
| 89 | 93 | A5 | |
| 8E | 9B | BD | |

| | | | |
|---|---|---|---|
| 91 | A1 | CF | |
| 8C | 99 | B7 | |
| 90 | 9F | C9 | |
| 8B | 97 | B1 | |

| | | | |
|---|---|---|---|
| 8C | 98 | AC | |
| 92 | A2 | CB | |
| 8D | 9A | B3 | |
| 93 | A4 | D3 | |

| | | | |
|---|---|---|---|
| 95 | A7 | DB | |
| 8F | 9D | BB | |
| 96 | A9 | E3 | |
| 90 | 9F | C3 | |

| | | | |
|---|---|---|---|
| 8E | 9B | B7 | |
| 94 | A5 | D7 | |
| 8C | 99 | AF | |
| 92 | A3 | CF | |

| | | | |
|---|---|---|---|
| 97 | AA | E7 | |
| 91 | A0 | C7 | |
| 95 | A8 | DF | |
| 8F | 9E | BF | |

| | | | |
|---|---|---|---|
| 8A | 92 | A1 | |
| 8E | 99 | B4 | |
| 8B | 93 | A5 | |
| 8F | 9B | B9 | |

| | | | |
|---|---|---|---|
| 90 | 9D | BE | |
| 8C | 95 | AA | |
| 91 | 9F | C3 | |
| 8D | 97 | AF | |

| | | | |
|---|---|---|---|
| 8B | 94 | A8 | |
| 8F | 9C | BB | |
| 8A | 92 | A3 | |
| 8E | 9A | B6 | |

| | | | |
|---|---|---|---|
| 91 | A0 | C5 | |
| 8D | 98 | B2 | |
| 90 | 9E | C0 | |
| 8C | 96 | AD | |

| | | | |
|---|---|---|---|
| 8D | 98 | AA | |
| 92 | A1 | C3 | |
| 8E | 9A | B0 | |
| 93 | A3 | C9 | |

| | | | |
|---|---|---|---|
| 95 | A5 | D0 | |
| 8F | 9C | B6 | |
| 96 | A7 | D6 | |
| 91 | 9E | BD | |

| | | | |
|---|---|---|---|
| 8F | 9B | B3 | |
| 94 | A4 | CD | |
| 8D | 99 | AD | |
| 93 | A2 | C6 | |

| | | | |
|---|---|---|---|
| 97 | A8 | D9 | |
| 91 | 9F | C0 | |
| 95 | A6 | D3 | |
| 90 | 9D | B9 | |

| | | | |
|---|---|---|---|
| 8B | 92 | A0 | |
| 8E | 99 | AF | |
| 8B | 93 | A3 | |
| 8F | 9A | B2 | |

| | | | |
|---|---|---|---|
| 90 | 9C | B6 | |
| 8C | 95 | A7 | |
| 91 | 9E | BA | |
| 8D | 97 | AB | |

| | | | |
|---|---|---|---|
| 8C | 94 | A5 | |
| 8F | 9B | B4 | |
| 8B | 92 | A1 | |
| 8E | 99 | B0 | |

| | | | |
|---|---|---|---|
| 91 | 9F | BC | |
| 8E | 98 | AD | |
| 90 | 9D | B8 | |
| 8D | 96 | A9 | |

| | | | |
|---|---|---|---|
| 8E | 98 | A8 | |
| 93 | A0 | BC | |
| 8F | 9A | AD | |
| 94 | A2 | C1 | |

| | | | |
|---|---|---|---|
| 95 | A4 | C6 | |
| 90 | 9C | B2 | |
| 96 | A6 | CB | |
| 91 | 9E | B7 | |

| | | | |
|---|---|---|---|
| 8F | 9B | AF | |
| 94 | A3 | C3 | |
| 8E | 99 | AA | |
| 93 | A1 | BE | |

| | | | |
|---|---|---|---|
| 97 | A7 | CD | |
| 92 | 9F | B9 | |
| 96 | A5 | C8 | |
| 91 | 9D | B4 | |

| | | | |
|---|---|---|---|
| 8C | 92 | 9E | |
| 8F | 98 | A9 | |
| 8C | 93 | A0 | |
| 8F | 99 | AB | |

| | | | |
|---|---|---|---|
| 90 | 9B | AE | |
| 8D | 95 | A3 | |
| 91 | 9C | B1 | |
| 8E | 96 | A6 | |

| | | | |
|---|---|---|---|
| 8D | 94 | A2 | |
| 90 | 9A | AD | |
| 8C | 92 | 9F | |
| 8F | 98 | AA | |

| | | | |
|---|---|---|---|
| 91 | 9D | B2 | |
| 8E | 97 | A7 | |
| 90 | 9B | AF | |
| 8D | 95 | A4 | |

| | | | |
|---|---|---|---|
| 8F | 98 | A6 | |
| 93 | 9F | B5 | |
| 90 | 99 | A9 | |
| 94 | A0 | B8 | |

| | | | |
|---|---|---|---|
| 95 | A2 | BC | |
| 91 | 9B | AD | |
| 96 | A4 | C0 | |
| 92 | 9D | B1 | |

| | | | |
|---|---|---|---|
| 90 | 9A | AB | |
| 95 | A1 | BA | |
| 8F | 98 | A7 | |
| 94 | 9F | B6 | |

| | | | |
|---|---|---|---|
| 97 | A5 | C2 | |
| 92 | 9E | B3 | |
| 96 | A3 | BE | |
| 91 | 9C | AF | |

| | | | |
|---|---|---|---|
| 8D | 92 | 9C | |
| 8F | 97 | A4 | |
| 8D | 93 | 9E | |
| 90 | 98 | A6 | |

| | | | |
|---|---|---|---|
| 90 | 99 | A8 | |
| 8E | 94 | A0 | |
| 91 | 9A | AA | |
| 8E | 95 | A2 | |

| | | | |
|---|---|---|---|
| 8D | 93 | 9F | |
| 90 | 98 | A7 | |
| 8D | 92 | 9D | |
| 8F | 97 | A5 | |

| | | | |
|---|---|---|---|
| 91 | 9B | AB | |
| 8F | 96 | A3 | |
| 91 | 9A | A9 | |
| 8E | 95 | A1 | |

| | | | |
|---|---|---|---|
| 90 | 98 | A4 | |
| 94 | 9E | AF | |
| 91 | 99 | A6 | |
| 95 | 9F | B1 | |

| | | | |
|---|---|---|---|
| 96 | A1 | B4 | |
| 92 | 9B | A9 | |
| 97 | A2 | B7 | |
| 93 | 9C | AC | |

| | | | |
|---|---|---|---|
| 91 | 9A | A8 | |
| 95 | A0 | B3 | |
| 90 | 98 | A5 | |
| 94 | 9E | B0 | |

| | | | |
|---|---|---|---|
| 97 | A3 | B8 | |
| 93 | 9D | AD | |
| 96 | A1 | B5 | |
| 92 | 9B | AA | |

| | | | |
|---|---|---|---|
| 8E | 92 | 9B | |
| 90 | 96 | A0 | |
| 8E | 93 | 9C | |
| 90 | 97 | A1 | |

| | | | |
|---|---|---|---|
| 91 | 98 | A2 | |
| 8F | 94 | 9D | |
| 91 | 99 | A3 | |
| 8F | 95 | 9E | |

| | | | |
|---|---|---|---|
| 8E | 93 | 9C | |
| 90 | 98 | A1 | |
| 8E | 92 | 9B | |
| 90 | 97 | A0 | |

| | | | |
|---|---|---|---|
| 91 | 9A | A4 | |
| 8F | 95 | 9F | |
| 91 | 99 | A3 | |
| 8F | 94 | 9E | |

| | | | |
|---|---|---|---|
| 92 | 98 | A1 | |
| 95 | 9C | A9 | |
| 92 | 99 | A3 | |
| 95 | 9D | AB | |

| | | | |
|---|---|---|---|
| 96 | 9E | AD | |
| 93 | 9A | A5 | |
| 97 | 9F | AF | |
| 94 | 9B | A7 | |

| | | | |
|---|---|---|---|
| 93 | 99 | A4 | |
| 96 | 9E | AC | |
| 92 | 98 | A2 | |
| 95 | 9D | AA | |

| | | | |
|---|---|---|---|
| 97 | A0 | B0 | |
| 94 | 9B | A8 | |
| 96 | 9F | AE | |
| 93 | 9A | A6 | |

| | | | |
|---|---|---|---|
| 8F 92 99 | 91 97 9E | 8F 93 9A | 91 98 9F |
| 90 95 9C | 8F 93 9A | 91 96 9D | 90 95 9C |
| 8F 92 99 | 91 98 9F | 8F 92 99 | 91 97 9E |
| 90 96 9D | 90 94 9B | 90 95 9C | 8F 94 9B |

| | | | |
|---|---|---|---|
| 94 98 9D | 97 9B A5 | 94 98 9F | 97 9C A7 |
| 96 9A A2 | 95 99 9F | 96 9B A4 | 95 9A A1 |
| 94 98 9E | 97 9C A6 | 94 98 9D | 97 9C A5 |
| 96 9B A3 | 95 99 A1 | 96 9A A3 | 95 99 A0 |

| | | | |
|---|---|---|---|
| 90 92 97 | 91 95 99 | 90 92 97 | 91 96 99 |
| 91 94 98 | 90 93 97 | 91 95 99 | 90 94 98 |
| 90 92 97 | 91 96 99 | 90 92 97 | 91 96 99 |
| 91 95 98 | 90 93 98 | 91 94 98 | 90 93 97 |

| | | | |
|---|---|---|---|
| 96 98 99 | 97 98 9E | 96 98 9A | 97 98 9F |
| 97 98 9C | 96 98 9A | 97 98 9D | 96 98 9C |
| 96 98 99 | 97 98 9F | 96 98 99 | 97 98 9E |
| 97 98 9D | 96 98 9B | 97 98 9C | 96 98 9B |

Highest contrast

| | | | |
|---|---|---|---|
| 92 92 92 | 92 92 92 | 92 92 92 | 92 92 92 |
| 92 92 92 | 92 92 92 | 92 92 92 | 92 92 92 |
| 92 92 92 | 92 92 92 | 92 92 92 | 92 92 92 |
| 92 92 92 | 92 92 92 | 92 92 92 | 92 92 92 |

| | | | |
|---|---|---|---|
| 98 98 98 | 98 98 98 | 98 98 98 | 98 98 98 |
| 98 98 98 | 98 98 98 | 98 98 98 | 98 98 98 |
| 98 98 98 | 98 98 98 | 98 98 98 | 98 98 98 |
| 98 98 98 | 98 98 98 | 98 98 98 | 98 98 98 |

Example with values from set 1 (the output values are the GB shades with 3=black, 0 = white):

| | |
|---|---|
| In (00h-FFh) | 000011112222333344445555666677778888999AAAABBBBCCCCDDDDEEEEFFFF |
| Min: 80 8F D0 | 333333333333333333333333333333332222111111111111111000000000000 |
| Def: 89 92 A2 | 333333333333333333333333333333332211110000000000000000000000000 |
| Max: 92 92 92 | 333333333333333333333333333333333000000000000000000000000000000 |

## 2.3 Game Boy Camera timings

The capture process is started when the A000 register of the Game Boy Camera cartridge is written any value with bit 0 set to '1'.

The Game Boy Camera cartridge is one of the few cartridges that use the PHI signal (clock from the GB). That signal is a 1MHz clock (1048576 Hz). The M64282FP chip needs a clock input too, which is half the frequency of the PHI pin (0.5 MHz, 524288Hz). The reason for that is that the sensor chip sometimes handles the signals on the rising edge of the clock but other times on the falling edge. NOTE: This means that the GB Camera shouldn't be used in GBC double speed mode!

The time needed to capture and process an image depends on the exposure time and the value of the N bit of the register 1 of the M64282FP chip.

In GAME BOY CYCLES (1MHz):

```
N_bit    = ([A001] & BIT(7)) ? 0 : 512
exposure = ([A002]<<8) | [A003]
CYCLES   = 32446 + N_bit + 16 * exposure
```

Divide that values by 2 to get the sensor clocks.

### 2.3.1 Capture process timings

The next values are in sensor clocks. Multiply by 2 to get Game Boy cycles.

```
- Reset pulse.
- Configure sensor registers.     (11 x 8 CLKs)
- Wait                                (1 CLK)
- Start pulse                         (1 CLK)
- Exposure time       (exposure_steps x 8 CLKs)
- Wait                               (2 CLKs)
- Read start
- Read period        (N=1 ? 16128 : 16384 CLKs)
- Read end
- Wait                                (3 CLKs)
- Reset pulse to stop the sensor

(88 + 1 + 1 + 2 + 16128 + 3 = 16223)

CLKs = 16223 + ( N_bit ? 0 : 256 ) + 8 * exposure
```

Obviously, that's the previous result divided by 2.

During the read process every pixel is written when it is read from the sensor. If the read process is stopped (by shutting the GB down, for example) the RAM will have the contents of the current picture until the read was stopped, from there it will have the data from the image captured before that one. The sensor transfers 128×128 pixels, but the upper and lower rows are corrupted. The Game Boy Camera controller only uses the medium rows of the sensor image. This means that it ignores the first 8 rows and the last 8 rows.

The clock signal during read period must be the same as the one used during the exposure time. If the clock during the read period is too slow the sensor will continue increasing the charge values of each pixel so the image will appear to be taken with a higher exposure time. The brightness doesn't seem to increase always, there seems to be some kind of limit.

# 3 The Game Boy Camera sensor (M64282FP)

The M64282FP does some processing to the captured image. First it performs an edge control, then it does gain control and last it does level control. The resulting analog value is the one that can be read in Vout pin. The sensor can capture infrared radiation, so images can be a bit strange compared to others captured by better sensors.

## 3.1 The M64282FP registers

### 3.1.1 Register 1

This corresponds to the register A001 of the Game Boy Camera. When shooting, the values change based on how much light there is.

| Symbol | Bits | Operation |
|---|---|---|
| N | 7 | Exclusively set vertical edge enhancement mode. |
| VH | 5-6 | Select vertical/horizontal edge operation mode. |
| G | 0-4 | Analog output gain. |

| G3 | G2 | G1 | G0 | Gain |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 14.0 |
| 0 | 0 | 0 | 1 | 15.5 |
| 0 | 0 | 1 | 0 | 17.0 |
| 0 | 0 | 1 | 1 | 18.5 |
| 0 | 1 | 0 | 0 | 20.0 |
| 0 | 1 | 0 | 1 | 21.5 |
| 0 | 1 | 1 | 0 | 23.0 |
| 0 | 1 | 1 | 1 | 24.5 |
| 1 | 0 | 0 | 0 | 26.0 |
| 1 | 0 | 0 | 1 | 29.0 |
| 1 | 0 | 1 | 0 | 32.0 |
| 1 | 0 | 1 | 1 | 35.0 |
| 1 | 1 | 0 | 0 | 38.0 |
| 1 | 1 | 0 | 1 | 41.0 |
| 1 | 1 | 1 | 0 | 45.5 |
| 1 | 1 | 1 | 1 | 51.5 |

If G4='1' the total gain is the previous one plus 6dB. The Game Boy Camera uses 00h, 04h, 08h and 0Ah. They are 14.0dB, 20.0dB, 26.0dB and 32dB, which translate to a gain of 5.01, 10.00, 19.95 and 39.81. The Game Boy Camera seems to like to duplicate the gain in each step.

### 3.1.2 Registers 2 and 3

They contain the exposure time (16 bit unsigned value). According to the M64282FP datasheet each step is 16 µs. In the GB it needs 16 PHI clocks for every step. If N='1' exposure_steps should be greater or equal than 0030h.

```
u16 exposure_steps = ([Reg2]<<8) | [Reg3]
Step time = 1 / 1048576 Hz * 16 = 0,954 µs * 16 = 15,259 µs
```

It's a bit less than the 16 µs the datasheet says, but it's close enough. Some example values to get acceptable pictures under various light conditions:

| Value | Conditions |
|---|---|
| 0030h | Objects under direct sunlight. |
| 0300h | Objects not under direct sunlight. |
| 0800h | Room during the day with good light. |
| 2C00h | Room at night with light. |
| 5000h | Room at night with no light, only a reading lamp. |
| F000h | Room at night with only a TV on in the background. |

Those values are illustrative, and they are important because the exposure time affects the time that is needed to take a picture. The initial value set by the ROM is 1000h.

### 3.1.3 Registers 4, 5 and 6

They are used by the 1-D processing kernel. They can't be set to any value from the GB Camera cartridge, only to 3 defined sets of values. The value of X must be fixed to 01h according to the datasheet. The values of P and M can't be modified if N='1', they are set to automatic values.

| [A000] | [4]=P | [5]=M | [6]=X | Description |
|---|---|---|---|---|
| 1(001) | 00h | 01h | 01h | Negative |
| 3(011) | 01h | 00h | 01h | Positive. This is the value the GB Camera ROM uses. |
| 5(101) | 01h | 02h | 01h | Used for edge detection |
| 7(111) | 01h | 02h | 01h | Used for edge detection |

### 3.1.4 Register 7

This corresponds to the register A004 of the Game Boy Camera.

| Symbol | Bits | Operation |
|---|---|---|
| E | 4-7 | Edge enhancement ratio |
| I | 3 | Select inverted/non-inverted output |
| V | 0-2 | Output node bias voltage (Vref) |

| E2 | E1 | E0 | Edge Enhancement Ratio |
|---|---|---|---|
| 0 | 0 | 0 | 50% |
| 0 | 0 | 1 | 75% |
| 0 | 1 | 0 | 100% |
| 0 | 1 | 1 | 125% |
| 1 | 0 | 0 | 200% |
| 1 | 0 | 1 | 300% |
| 1 | 1 | 0 | 400% |
| 1 | 1 | 1 | 500% |

| V2 | V1 | V0 | Vref(V) |
|---|---|---|---|
| 0 | 0 | 0 | 0.0 |
| 0 | 0 | 1 | 0.5 |
| 0 | 1 | 0 | 1.0 |
| 0 | 1 | 1 | 1.5 |
| 1 | 0 | 0 | 2.0 |
| 1 | 0 | 1 | 2.5 |
| 1 | 1 | 0 | 3.0 |
| 1 | 1 | 1 | 3.5 |

### 3.1.5 Register 0

This corresponds to the register A005 of the Game Boy Camera.

| Symbol | Bits | Operation |
|---|---|---|
| Z | 6-7 | Zero point calibration (Set dark level output signal to Vref) |
| O | 0-5 | Output reference voltage (In both plus and minus direction) |

The two calibration modes that are used are '10' (calibration for positive signal) and '00' (disabled). Note: The undefined Z='11' configuration is the same as Z='00'. The unused configuration Z='01' is used for calibration for negative signal.

The reference voltage is adjusted in 32mV steps, with the most significant bit being the sign (O5='1' is positive). The output reference voltage is calculated like this:

```
int reg_offset = REG[0] & 0x1F;
float offset_step = ( (reg_offset & 0x20) ? 0.032 : -0.032 )
float OFFSET = offset_step * (reg_offset & 0x1F)
```

## 3.2 M64282FP image processing

The image processing is done in three stages: Edge control, gain control and level control. Since the sensor is very badly documented, most of this part is guesswork (specially gain and level control). Anyway, only the edge control is needed to emulate the Game Boy Camera.

```
Original                [0V,5V]
Invert                  [0V,5V] \
3x3 kernel              [0V,5V]  | Edge control
1-D filter              [0V,5V] /
Zero calibration        [0V,5V] \
Offset                  [0V,5V]  | Gain control
Gain -> Origin is 2.5 V [0V,5V] /
Vref                    [0V,5V] -> Level control
Final clamped output    [0.9V, 4.0V]
```

### 3.2.1 Edge control

Note that, before any edge processing, the inverting bit of register 7 is processed. The edge processing type depends on the value of registers N, VH and E. If 1-D filtering is enabled, the registers P, M and X are also used. If N is set to '1' the registers P and M are ignored as they are used by the 3×3 filtering kernel (so changing the value used to trigger the capture doesn't change the resulting image). When using edge extraction modes Z1 should be set to '0'. In any other cases, it should be set to '1'. The 3×3 filtering kernel is applied before the 1-D filtering.

| N | VH1 | VH0 | E3 | Description | 1-D enabled | 3×3 filtering matrix kernel |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Positive image | Yes | Disabled. |
| 0 | 0 | 1 | 0 | Horiz. enhancement | Yes | 0 0 0 \| -1 3 -1 \| 0 0 0 |
| 0 | 0 | 1 | 1 | Horiz. extraction | Yes | 0 0 0 \| -1 2 -1 \| 0 0 0 |
| 1 | 1 | 0 | 0 | Vert. enhancement | No | 0 -1 0 \| 0 3 0 \| 0 -1 0 |
| 1 | 1 | 0 | 1 | Vert. extraction | No | 0 -1 0 \| 0 2 0 \| 0 -1 0 |
| 1 | 1 | 1 | 0 | 2D enhancement | No | 0 -1 0 \| -1 5 -1 \| 0 -1 0 |
| 1 | 1 | 1 | 1 | 2D extraction | No | 0 -1 0 \| -1 4 -1 \| 0 -1 0 |

*Table 1: Edge processing modes.*

Other combinations are undefined and shouldn't be used.

- **3×3 filtering matrix kernel**

This consists on a 3×3 matrix. The configuration values to use each mode and the resulting values used to calculate the final image are specified in 1. The following expressions are used to calculate the result (α: edge enhancement ratio), and the images show examples for α = 100%:

| Edge mode | Operation |
|---|---|
| Vertical edge extraction | {2P-(MN+MS)} × α |
| Horizontal edge extraction | {2P-(MW+ME)} × α |
| 2D edge extraction | {4P-(MN+MS+ME+MW)} × α |
| Vertical edge enhancement | P + {2P-(MN+MS)} × α |
| Horizontal edge enhancement | P + {2P-(MW+ME)} × α |
| 2D edge enhancement | P + {4P-(MN+MS+ME+MW)} × α |

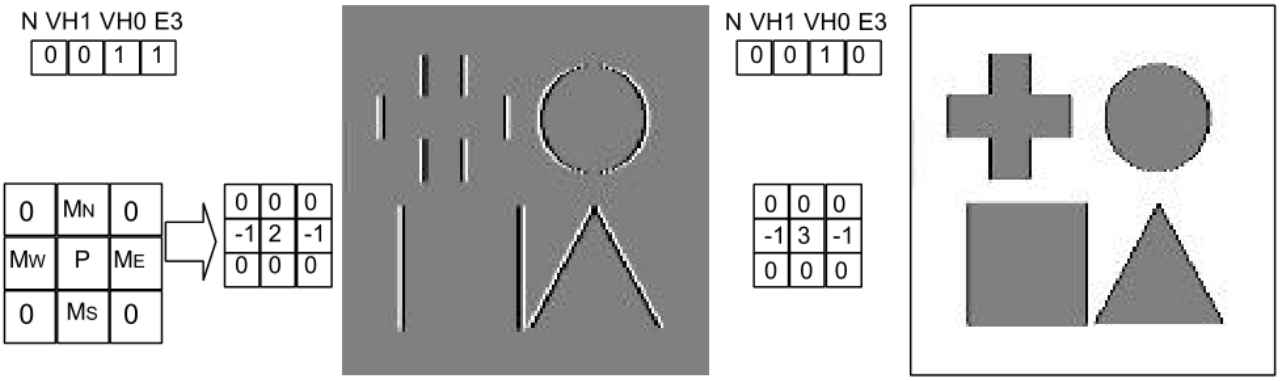


*Figure 1: Horizontal edge processing modes.*

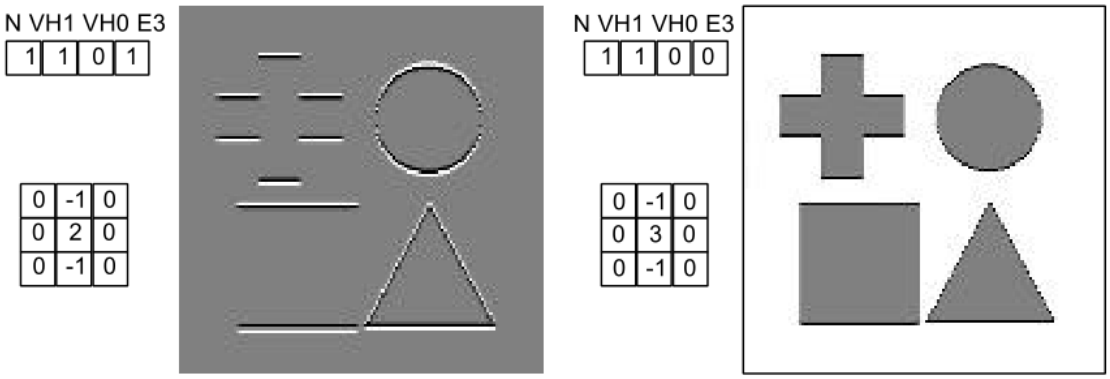


*Figure 2: Vertical edge processing modes.*

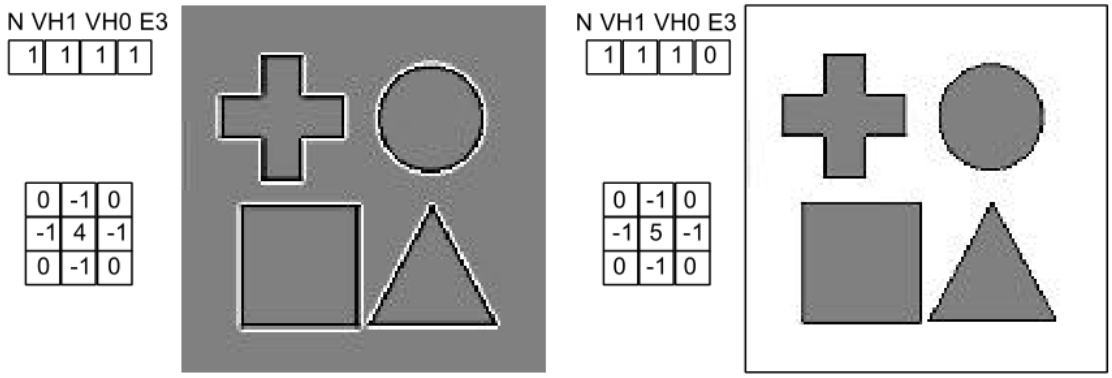


*Figure 3: 2D edge processing modes.*

- **1-D filtering mode**

When using edge extraction, Z1 should be '0'. It uses the P, M and X registers like this:

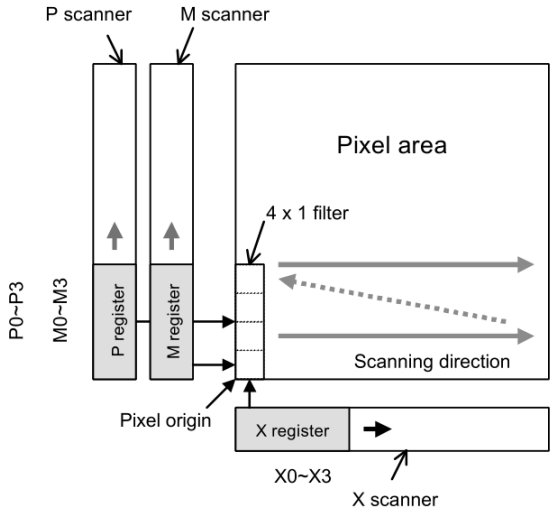


*Figure 4: 1-D filtering hardware.*

P, M and X are bitmasks. The bits set to '1' select lines of the image. The lines selected in P register (plus?) are added, the ones in M register (minus?) are subtracted. If the 2 bits of the same line are set to '1' the result is the same as if they were '0'. According to the datasheet, the X register must be set to '01h', the Game Boy Camera does it automatically. The three possible configurations depending on the value written to A000 register when triggering the capture are:

| [A000] | [4]=P | [5]=M | [6]=X | Description |
|---|---|---|---|---|
| 1(001) | 00h | 01h | 01h | Negative. Figure 6. |
| 3(011) | 01h | 00h | 01h | Positive. Figure 5. |
| 5(101) | 01h | 02h | 01h | Used for edge extraction. Figure 7. |
| 7(111) | 01h | 02h | 01h | Used for edge extraction. Figure 7. |

The three following images show how it works:

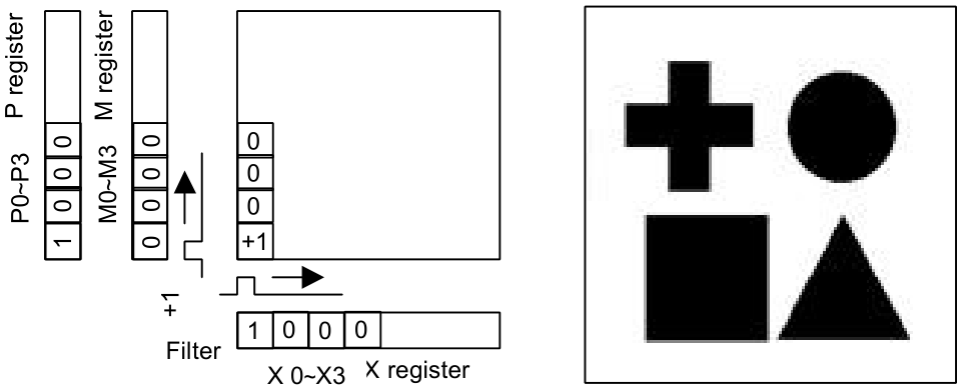


*Figure 5: Positive image.*

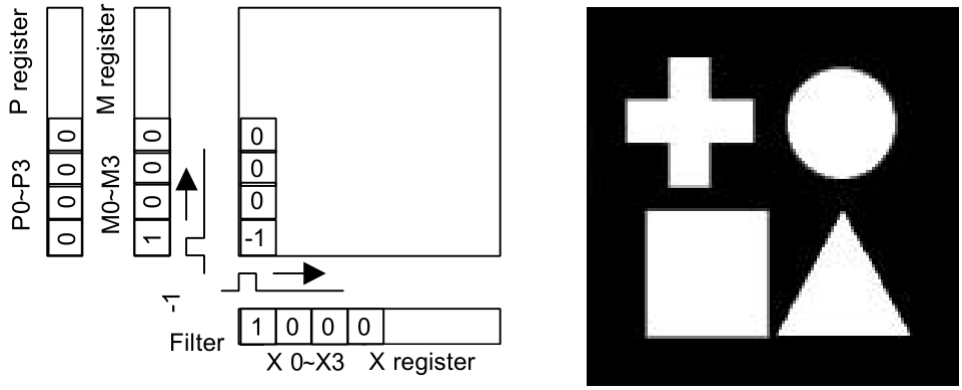


*Figure 6: Negative image.*

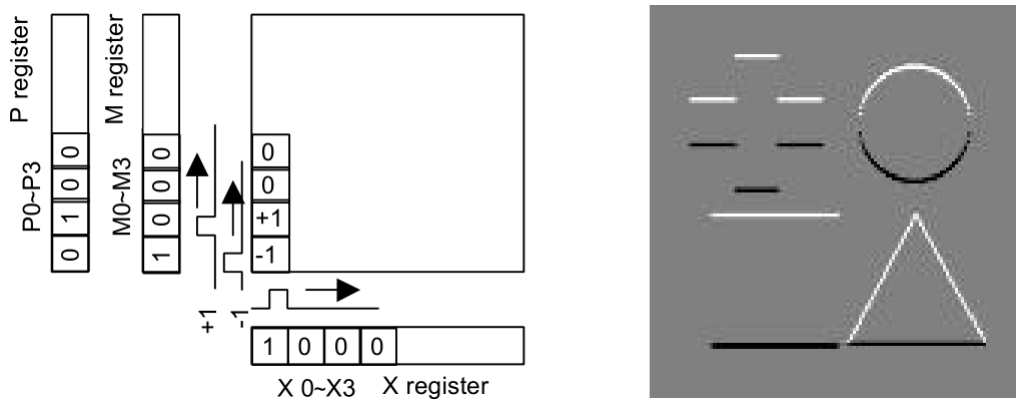


*Figure 7: Edge extraction.*

### 3.2.2 Gain control

This is not explained properly in the datasheet, so don't trust this information too much. In this phase the zero point calibration is applied (at the beginning?), then the output reference voltage is added and at last the gain is applied.

### 3.2.3 Level control

Vref is applied and the output is clamped to [0.9V, 4.0V].

# 4 Sample code for emulators

The following code is used to convert a greyscale image to the Game Boy Camera format. GB_CameraTakePicture() should be called when bit 0 of A000 register is st to '1'. The emulator should wait CAM_CLOCKS_LEFT until the bit 0 is cleared. The gain and level control are not needed to emulate the Game Boy Camera because webcams do that automatically. In fact, trying to emulate that will probably break the image. The code is not very clean because it has been extracted from my GiiBiiAdvance, but it seems to handle all used configurations of edge handling.

Note that the actual Game Boy Camera sensor is affected by infrared so the emulation can't be perfect anyway. A good way of converting a RGB image into grayscale is to do:

$$V = (2 \times R + 5 \times G + 1 \times B) / 8$$

```
// The actual sensor image is 128x126 or so.
#define GBCAM_SENSOR_EXTRA_LINES (8)
#define GBCAM_SENSOR_W (128)
#define GBCAM_SENSOR_H (112+GBCAM_SENSOR_EXTRA_LINES)

#define GBCAM_W (128)
#define GBCAM_H (112)

#define BIT(n) (1<<(n))

// Webcam image
static int gb_camera_webcam_output[GBCAM_SENSOR_W][GBCAM_SENSOR_H];
// Image processed by sensor chip
static int gb_cam_retina_output_buf[GBCAM_SENSOR_W][GBCAM_SENSOR_H];

//-------------------------------------------------------------------

static inline int clamp(int min, int value, int max)
{
    if(value < min) return min;
    if(value > max) return max;
    return value;
}

static inline int min(int a, int b) { return (a < b) ? a : b; }

static inline int max(int a, int b) { return (a > b) ? a : b; }

//-------------------------------------------------------------------

static inline u32 gb_cam_matrix_process(u32 value, u32 x, u32 y)
{
    x = x & 3;
    y = y & 3;

    int base = 6 + (y*4 + x) * 3;

    u32 r0 = CAM_REG[base+0];
    u32 r1 = CAM_REG[base+1];
    u32 r2 = CAM_REG[base+2];

    if(value < r0) return 0x00;
    else if(value < r1) return 0x40;
    else if(value < r2) return 0x80;
    return 0xC0;
}

static void GB_CameraTakePicture(void)
{
    int i, j;

    //-------------------------------------------------

    // Get webcam image
    // ----------------

    GB_CameraWebcamCapture();

    //-------------------------------------------------

    // Get configuration
    // -----------------

    // Register 0
    u32 P_bits = 0;
    u32 M_bits = 0;

    switch( (CAM_REG[0]>>1)&3 )
    {
        case 0: P_bits = 0x00; M_bits = 0x01; break;
        case 1: P_bits = 0x01; M_bits = 0x00; break;
        case 2: case 3: P_bits = 0x01; M_bits = 0x02; break;
        default: break;
    }

    // Register 1
    u32 N_bit = (CAM_REG[1] & BIT(7)) >> 7;
    u32 VH_bits = (CAM_REG[1] & (BIT(6)|BIT(5))) >> 5;

    // Registers 2 and 3
    u32 EXPOSURE_bits = CAM_REG[3] | (CAM_REG[2]<<8);
```

```
    // Register 4
    const float edge_ratio_lut[8] = { 0.50, 0.75, 1.00, 1.25, 2.00, 3.00, 4.00, 5.00 };

    float EDGE_alpha = edge_ratio_lut[(CAM_REG[4] & 0x70)>>4];

    u32 E3_bit = (CAM_REG[4] & BIT(7)) >> 7;
    u32 I_bit = (CAM_REG[4] & BIT(3)) >> 3;

    //-----------------------------------------------

    // Calculate timings
    // -----------------

    CAM_CLOCKS_LEFT = 4 * ( 32446 + ( N_bit ? 0 : 512 ) + 16 * EXPOSURE_bits );

    //-----------------------------------------------

    // Sensor handling
    // ---------------

    //Copy webcam buffer to sensor buffer applying color correction
    for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
    {
        int value = gb_camera_webcam_output[i][j];
        value = 128 + 0.75f*(float)(value-128);
        gb_cam_retina_output_buf[i][j] = clamp(0,value,255);
    }

    // Apply exposure time (with a reference of 0x0100)
    for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
    {
        int result = gb_cam_retina_output_buf[i][j];
        result = ( (result * EXPOSURE_bits ) / 0x0100 );
        gb_cam_retina_output_buf[i][j] = clamp(0,result,255);
    }

    if(I_bit) // Invert image
    {
        for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
            gb_cam_retina_output_buf[i][j] ^= 255;
    }

    int temp_buf[GBCAM_SENSOR_W][GBCAM_SENSOR_H];

    u32 filtering_mode = (N_bit<<3) | (VH_bits<<1) | E3_bit;
    switch(filtering_mode)
    {
        case 0x0: // 1-D filtering
        {
            for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
            {
                temp_buf[i][j] = gb_cam_retina_output_buf[i][j];
            }
            for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
            {
                int ms = temp_buf[i][min(j+1,GBCAM_SENSOR_H-1)];
                int px = temp_buf[i][j];

                int value = 0;
                if(P_bits&BIT(0)) value += px;
                if(P_bits&BIT(1)) value += ms;
                if(M_bits&BIT(0)) value -= px;
                if(M_bits&BIT(1)) value -= ms;
                gb_cam_retina_output_buf[i][j] = clamp(0,value,255);
            }
            break;
        }
        case 0x2: //1-D filtering + Horiz. enhancement : P + {2P-(MW+ME)} * alpha
        {
            for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
            {
                int mw = gb_cam_retina_output_buf[max(0,i-1)][j];
                int me = gb_cam_retina_output_buf[min(i+1,GBCAM_SENSOR_W-1)][j];
                int px = gb_cam_retina_output_buf[i][j];

                temp_buf[i][j] = clamp(0,px+((2*px-mw-me)*EDGE_alpha),255);
            }
            for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
            {
                int ms = temp_buf[i][min(j+1,GBCAM_SENSOR_H-1)];
                int px = temp_buf[i][j];
```

```
                int value = 0;
                if(P_bits&BIT(0)) value += px;
                if(P_bits&BIT(1)) value += ms;
                if(M_bits&BIT(0)) value -= px;
                if(M_bits&BIT(1)) value -= ms;
                gb_cam_retina_output_buf[i][j] = clamp(0,value,255);
            }
            break;
        }
        case 0xE: //2D enhancement : P + {4P-(MN+MS+ME+MW)} * alpha
        {
            for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
            {
                int ms = gb_cam_retina_output_buf[i][min(j+1,GBCAM_SENSOR_H-1)];
                int mn = gb_cam_retina_output_buf[i][max(0,j-1)];
                int mw = gb_cam_retina_output_buf[max(0,i-1)][j];
                int me = gb_cam_retina_output_buf[min(i+1,GBCAM_SENSOR_W-1)][j];
                int px  = gb_cam_retina_output_buf[i][j];

                temp_buf[i][j] = clamp(0,px+((4*px-mw-me-mn-ms)*EDGE_alpha),255);
            }
            for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
            {
                gb_cam_retina_output_buf[i][j] = temp_buf[i][j];
            }
            break;
        }
        case 0x1:
        {
            // In my GB Camera cartridge this is always the same color.
            // The datasheet of the sensor doesn't have this configuration
            // documented. Maybe this is a bug?
            for(i=0;i<GBCAM_SENSOR_W;i++) for(j=0;j<GBCAM_SENSOR_H;j++)
            {
                gb_cam_retina_output_buf[i][j] = 0x80;
            }
            break;
        }
        default:
        {
            // Ignore filtering
            printf("Unsupported GB Cam mode: 0x%X\n%02X %02X %02X %02X %02X %02X",
                   filtering_mode,
                   CAM_REG[0],CAM_REG[1],CAM_REG[2],CAM_REG[3],CAM_REG[4],CAM_REG[5]);
            break;
        }
    }

    //------------------------------------------------

    // Controller handling
    // -------------------

    int fourcolorsbuffer[GBCAM_W][GBCAM_H]; // buffer after controller matrix

    // Convert to Game Boy colors using the controller matrix
    for(i = 0; i < GBCAM_W; i++) for(j = 0; j < GBCAM_H; j++)
        fourcolorsbuffer[i][j] =
            gb_cam_matrix_process(
                gb_cam_retina_output_buf[i][j+(GBCAM_SENSOR_EXTRA_LINES/2)],i,j);

    // Convert to tiles
    u8 finalbuffer[14][16][16]; // final buffer
    memset(finalbuffer,0,sizeof(finalbuffer));
    for(i = 0; i < GBCAM_W; i++) for(j = 0; j < GBCAM_H; j++)
    {
        u8 outcolor = 3 - (fourcolorsbuffer[i][j] >> 6);

        u8 * tile_base = finalbuffer[j>>3][i>>3];
        tile_base = &tile_base[(j&7)*2];

        if(outcolor & 1) tile_base[0] |= 1<<(7-(7&i));
        if(outcolor & 2) tile_base[1] |= 1<<(7-(7&i));
    }

    // Copy to cart ram...
    memcpy(&(SRAM[0][0x0100]),finalbuffer,sizeof(finalbuffer));
}
```