Sega Genesis Lightguns

by Eke-Eke (29/07/2008)
http://gxdev.wordpress.com/

Introduction

The following document is a compilation of my notes about the Sega Menacer and Konami Justifier lightguns. This mainly covers undocumented aspect of these devices and is principally designed for emulator authors who wants to implement or improve accuracy of lightgun emulation.

Please note that **NONE** of the described features have been tested on real hardware, everything directly comes from analyzing and reverse engineering disassembled gun games ROM images. So, even if it seems to work well on emulators for the mentioned games, some behaviour descriptions or deductions might be incorrect. Feel free to contact me for any correction or comments.

Finally, before reading this document, you may want to consult the following publications, which explain important aspects of the Genesis/Megadrive hardware and might be necessary to understand some of the concepts exposed in this document:

- Sega Genesis hardware notes by Charles MacDonald (gen-hw.txt)
- Sega Genesis VDP documentation by Charles MacDonald (genvdp.txt)
- Sega Genesis I/O Chip & Peripherals by Charles MacDonald (gen_io.txt)

The following official documents could also be useful:

- Official Genesis Software Manual & Technical bulletins from Sega (http://www.spritesmind.net/_GenDev/forum/viewtopic.php?t=227)
- Sega Menacer patent (US5351969) http://v3.espacenet.com/textdraw?DB=EPODOC&IDX=US5351969&F=0&QPN=US5351969

I also would like to thanks Tasco Deluxe for providing the Sega Menacer US Patent number.

Lightgun games

On the Sega Genesis/Megadrive, the following games are known to have gun support:

- Sega Menacer :
  - Menacer 6-in-1 game pack
  - T2: Arcade Game
  - Body Count

- Konami Justifier :
  - Lethal Enforcers
  - Lethal Enforcers 2

Some Mega-CD games are also known (or rumoured) to have support either for Sega Menacer or Konami Justifier but I haven't actually been able to test them. This document only covers emulation of lightgun games for the Sega Genesis/Megadrive.

How lightgun works ?

All gun games require the lightgun to be connected on IO port#2 and most of them require a gamepad to be plugged in IO port #1. There is however no known hardware restriction to use them on port #1, it's just that no commercial games actually do this.

From software, IO port is accessed through the associated DATA register and configured through the CTRL register.

DATA 2 register ($A10005):

PD7		R/W
PD6	TH pin	R/W
PD5	TR pin	R/W
PD4	TL pin	R/W
PD3	“LEFT” pin	R/W
PD2	“RIGHT” pin	R/W
PD1	“UP” pin	R/W
PD0	“DOWN” pin	R/W

CTRL2 register ($A1000B):

PC7	HL output control	R/W
PC6	PD6 mode (0:input, 1:output)	R/W
PC5	PD5 mode (0:input, 1:output)	R/W
PC4	PD4 mode (0:input, 1:output)	R/W
PC3	PD3 mode (0:input, 1:output)	R/W
PC2	PD2 mode (0:input, 1:output)	R/W
PC1	PD1 mode (0:input, 1:output)	R/W
PC0	PD0 mode (0:input, 1:output)	R/W

(1) guns work by triggering a high to low transition on TH pin (bit6) when their lens sensor detects the pixel beam on TV, so:

- PD6 (TH) should be configured as an input to be able to detect Gun Position
- PD5 (TR) & PD4 (TL) are generally configured as outputs and are used for sending commands to the gun.
- PD3-PD0 are configured as inputs and usually return gun buttons status

(2) when bit7 is set, a high to low TH transition will trigger HL input on the VDP side:

- if bit 3 (IE2) of VDP register #11 is set, a level 2 interrupt is triggered.
- if bit 1 (M3) of VDP register #0 is set, HVC counter is freezed (latched) for later read.

All Genesis/Megadrive lightgun games enable INT2 triggering and read the **HV counter** in their interrupt handler. By comparison, on the Sega Master System (or Sega Genesis in SMS compatibility mode), no interrupts could be triggered: games usually have to poll the DATA port and wait for a high-to-low TH transition then read HV counter.

HV counter ($C00008):

D15-D8	current vertical line (V counter)
D7-D0	current horizontal pixel position (H counter)

During active display, V counter corresponds to the current drawn scanline: it starts with $00 and, depending on the height of the active display, ends with $DF (224 lines mode) or $EF (240 lines mode, PAL only). Other values are outside the active display range, which is no concern regarding how gun detection works.

According to Charles MacDonald who made tests on a real Sega Genesis, H counter values range is:

- ***H32 cell mode*** *(active display = 256 pixels out of 342 pixels)*
  0x00-0x93
  0xE9-0xFF

- ***H40 cell mode*** *(active display = 320 pixels out of 422 pixels)*
  0x00-0xB6
  0xE4-0xFF

- one H counter unit is equivalent to two pixels.

- the gap in the counter probably coincides with HSYNC period, when the electron beam returns to the left side of the TV screen.

- On a side note, it is currently not known if value 0x00 corresponds to the first *active* pixel (pixel 0) or to the first pixel of the left border (blanked).

- Some games enable HVC latch but some of them does not. In the later case, H counter will obviously return a higher value than the current beam position, due to hardware latency: games will have to take this in account in their gun positioning routine.

Sega Menacer

Technical description

This is based on the Sega Menacer patent description.

The lightgun is composed of 2 parts:

- a mobile unit (the gun), detecting raster lines on television thanks to an optoelectronic sensor
- a fixed unit (IR transceiver/receiver) plugged into the Sega Genesis IO port.

Both units are communicating through IR pulses:

1) the mobile unit is normally constantly transmitting a pulse for each detected scanline. As the gun lens "sees" a circular spot, a variable number of pulses is transmitted. A counter inside the fixed unit detects the received pulses and when the count reaches 8, it triggers the TH signal on the IO port. This is used to average the current Y position of the gun by notifying the game software only once time.

2) When buttons status is required, the fixed unit needs to send an IR sync pulse to the mobile receiver, which responds by a certain amount of pulse (0-15), corresponding to the state of the buttons (4 buttons are mapped). From the Sega Genesis point of view, PD0-PD3 of DATA register will exactly reflect this number of received pulses and the state of each four buttons as well.

The IR sync pulse is controlled by a RST signal, which in fact have two possible lengths:

- 2 or 3 microseconds: this is the "counter reset" signal, used to reset the above pulse counter. This is sent after receiving button data, at the end of VBLANK.

- 10 microseconds: this is the "main reset" signal, used to issue the IR sync pulse described above and start buttons data acquisition. This is sent at the beginning of VBLANK.

The RST signal is controlled by software using TH and TR pins (bit5 & bit4 of DATA register)

DATA register

PD7		
PD6	input	pixel beam detection (0: visible, 1: not visible)
	output	unused
PD5	input	unused (unknown value)
	output	RST signal level (0: enabled, 1:disabled)
PD4	input	unused (unknown value)
	output	RST signal length (0: short, counter reset, 1: long, main reset)
PD3	input	START button (0: released, 1: pressed)
	output	unused
PD2	input	C button (0: released, 1: pressed)
	output	unused
PD1	input	A button (0: released, 1: pressed)
	output	unused
PD0	input	B button (0: released, 1: pressed)
	output	unused

Buttons acquisition sequence

To get button status, PD5 & PD4 should be set as outputs and PD3-PD0 as inputs. Then the following sequence is done:

1) *"Main" Reset*

write IO PORT 2 DATA--> 0xff (00000328) 11
write IO PORT 2 DATA--> 0xdf (0000034E) 01
write IO PORT 2 DATA--> 0xff (0000035A) 11
write IO PORT 2 DATA--> 0xdf (00000364) 01

2) *read button status*

read IO PORT 2 DATA (00000372)

3) *"Counter" Reset*

write IO PORT 2 DATA--> 0xff (0000037A) 11
write IO PORT 2 DATA--> 0xcf (00000384) 00

Y Position

As explained above, only one level2 interrupt would happen on each frame, occurring on the line that approximately corresponds to the current gun Y position, preliminary approximation being done by the gun hardware.

a) to calculate Y position for the current frame, game softwares usually do some kind of movement interpolation. For example, *6-in-1 game* calculates the average V-Counter value from the last eight frames. *Body Count* and *T2: Arcade Game* seem to use a more complicated calculation method.

b) V counter value is directly converted into an Y coordinate.

c) depending on the game or where as gun calibration is possible, a vertical offset is added to the result: *Menacer 6-in-1 pack* uses **0** by default, *T2: Arcade Game* uses **8** by default and *Body Count* uses **16** by default. This is probably done to compensate the fact that the gun does not exactly “sees” 16 (8x2) lines, depending on the distance between the TV and the gun lens, the visible area is probably larger.

d) due to the internal counter inside Menacer hardware, no interrupts could theoretically be triggered on line ($00-$07), making the top eight lines area uncovered by the cursor.

X Position

a) contrary to V counter value, H counter value is converted into an X coordinate through a precalculated table stored in memory.

b) as for Y position, movement interpolation is done by the software as an average of the previous frame's X coordinates.

c) an offset is applied by default in the H counter table when converting H counter value into actual X coordinate. *6-in-1 pack* uses **$4E** as offset value and *Body Count* uses **$48**, where as *T2: Arcade Game* uses **$84**.

Notes:

- ✓ this is probably done to take the following hardware latencies in account:
  - o the gun lens "sees" a circular spot so the X centre position is some pixels away from the start of the detection pulse.
  - o there is a delay between detection pulse and TH transition, due to the lightgun hardware latency and IR transmissions.
  - o there is also a delay between TH transition and INT2 triggering or/and HVC latch, due to I/O and VDP chips latencies.
  - o finally, I’m not sure if value 0x00 really corresponds to the first pixel of the *active* display area. It might be possible that H counter starts with the first pixel of the left blanked border area.
- ✓ *T2:Arcade Game* routine does *NOT* set VDP register #0 bit1 (M3) so H counter will NOT be latched. This could explain why the gun routine in *T2: Arcade Game* uses a larger offset in order to take interrupt level 2 acceptance timing in account.
- ✓ *6-in-1 pack* shifts the cursor position for **8 pixels** to the left after calculating the interpolated position. This means the last 8 pixels display area cannot be covered by the cursor and that the cursor starts moving eight pixels before the start of the active display area. Depending on the size of the drawn cursor, it will remains invisible or only partially visible. On a similar way, *Body Count* reduced the active display range, the left and right **8 pixels** being inactive.

Here is the H counter range and the corresponding cursor position on screen. All Genesis games use H40 cell mode, other games (such as Sega-CD ones) may use H32 cell mode.

Menacer 6-in-1 pack

$44-$4D	BLANK: cursor is not visible
$4E-$51	LEFT BORDER: cursor is moving but not fully visible
$52-$B6; $E5-$FF; $00-$0C	ACTIVE DISPLAY: cursor is visible & moving
$0D-$43	RIGHT BORDER: cursor is fixed but visible

T2: Arcade Game

$70-$83	LEFT BORDER: cursor is fixed
$84-$B6; $E5-$FF; $00-$42	ACTIVE DISPLAY: cursor is visible & moving
$43-$6F	RIGHT BORDER: cursor is fixed

BodyCount

$35-$47	LEFT BORDER: cursor is fixed
$48-$B6; $E5-$FF;	ACTIVE DISPLAY: cursor is visible & moving
$00-$34	RIGHT BORDER: cursor is fixed

- ✓ H counter table values are scaled up, which explains why the cursor "active" range is only 290 pixel counts wide, which is smaller than the 320 pixels active display area. This is probably done to take the fastest pixel clock (used in H40 mode) in account.

- ✓ H counter gap is correctly taken in account but is 2 pixels off (H counter restarts at $E5 instead of $E4). Note that the routine that set the internal H counter table (can be found at offset $315FA in disassembled Menacer ROM) explicitly uses $B6 and $E5 limit values.

From above observations, I found out that a very accurate way to emulate the H counter latch value was to return something like this:

```
hc_latch = hc_table[((gun.x * 290) / 320 / 2) + gun.x_offset) % 211]
```

with:

- gun.x_offset = 0x44 (Body Count) , 0x52 (Menacer) or 0x84 (T2: Arcade Game)
- gun.x being the current cursor position (0-320)
- hc_table being a table with H counter values as described in introduction.

However, if you do not plan to draw your own cursor, “perfect” positioning is not required and the latency could be accurately emulated instead of using specific game offsets.

Konami Justifier

The "Blue" Gun is connected to the Sega Genesis IO port #2. A "Pink" gun can also be connected to the first gun. A gamepad is required to be plugged into IO port #1. I have actually no informations about how the gun hardware is internally working.

DATA register

PD7		
PD6	input	pixel beam detection (0: visible, 1: not visible)
	output	gun detection (see below, 0: normal, 1: force button state)
PD5	input	unused (see below, always return 1)
	output	gun selection (0: “blue” gun, 1: “pink” gun)
PD4	input	unused (see below, always return 1)
	output	gun input switch (0: enabled, 1: disabled/reset)
PD3	input	unused (see below, always return 0)
	output	unused
PD2	input	unused (see below, always return 0)
	output	unused
PD1	input	START button (0:pressed, 1:released)
	output	unused
PD0	input	Trigger button (0:pressed, 1:released)
	output	unused

Gun detection routine

Games software execute a commonly used SEGA detection routine, which calculates a 4-bits ID code to determine the type of the connected device. To get the ID code, TH is set to output mode (CTRL2 = $40) and the routine alternately set TH=1 then TH=0.

The following logical operation is used to determine each four bits of the ID code:

```
ID3 = (TH == 1) & (DATA bit3 | DATA bit2)
ID2 = (TH == 1) & (DATA bit1 | DATA bit0)
ID1 = (TH == 0) & (DATA bit3 | DATA bit2)
ID0 = (TH == 0) & (DATA bit1 | DATA bit0)
```

- ✓ a gamepad (6-buttons or 3-buttons) will return ID=0xD
- ✓ Konami Justifier is expected to return ID=0x1

This means the Justifier should return the following byte values when TH is set as output:

- if TH = 1: DATA = ?1110000
- if TH = 0: DATA = ?01100xx with xx different from 00

I'm not sure why Justifier returns such values, maybe setting TH=1 will force input lines to 0.

Notes:

- ✓ bit2 (LEFT pin) & bit3 (RIGHT pin) maybe always return 0 (note that those bits are unused later).

- ✓ even if Justifier is properly detected, the game will once try to read DATA2 register, as if a gamepad was plugged. I don't know if this is an emulation timing issue but if PD5 (TL pin) or PD4 (TR pin) return 0, the game will assume either START or A button as been pressed and will directly jump into the game. It's safer to assumed those bits always return 1 when set as inputs.

Button Acquisition

To get buttons status, PD5 & PD4 should be set as outputs and PD1-PD0 should be set as inputs.

The following sequence is executed at the start of each VBLANK, twice repeated to get buttons status for the two guns:

1) gun acquisition mode is enabled:
   - bit5 of DATA port is set to select the appropriate gun
   - bit4 of DATA port is cleared to enable acquisition

2) buttons status is read on DATA port

3) gun is reseted (disabled):
   - bit5 of DATA port is set to select the appropriate gun
   - bit4 of DATA port is set to reset/disable acquisition

This has been deducted from analysing the VINT gun routine in Lethal Enforcers:

```
226(110486): write IO PORT 2 DATA--> 0x00  (00000908)
226(110514): read  IO PORT 2 DATA          (00000946)
226(110530): read  IO PORT 2 DATA          (00000946)
226(110570): write IO PORT 2 DATA--> 0x10  (0000092A)
226(110598): write IO PORT 2 DATA--> 0x20  (00000936)
226(110626): read  IO PORT 2 DATA          (00000946)
226(110642): read  IO PORT 2 DATA          (00000946)
226(110682): write IO PORT 2 DATA--> 0x30  (00000958)
```

Gun Position

To get gun position, CTRL2 register bit7 should be set to enable HL output triggering and TH should be set as an input.

The following sequence is executed at the start of each frame (approx. on line 0), alternately for each gun:

1) gun is reseted (disabled):

    - bit5 of DATA port is set to select the appropriate gun
    - bit4 of DATA port is set to reset/disable acquisition

2) gun acquisition mode is enabled:

    - bit5 of DATA port is set to select the appropriate gun
    - bit4 of DATA port is cleared to enable acquisition

This has been deducted from analysing the VINT gun routine in Lethal Enforcers:

```
(frame N)
 0(376): write IO PORT 2 DATA--> 0x10(00000A86)
 0(406): write IO PORT 2 DATA--> 0x0  (00000A90)

(frame N+1)
 0(386): write IO PORT 2 DATA--> 0x30(00000A86)
 0(416): write IO PORT 2 DATA--> 0x20(00000A90)
```

Note:

Both games give the player the ability to calibrate the guns. Basically, the game will ask you to aim at the centre of the screen then it reads the HV counter and calculates H & V offsets that should afterward be applied on the H and V counters value each time they are read in the level 2 interrupt handler.

X Position

Both games use the H32 cell mode (256 pixels active area). By default, the following H counter ranges are used:

Lethal Enforcers

$00-$7F	ACTIVE DISPLAY: cursor is visible & moving
$80-$FF	INACTIVE

This game does **NOT** latch HV counter and does **NOT** use any X offset by default to take hardware latency in consideration. This would mean that, on real hardware, gun calibration is highly advised

Lethal Enforcers II

$18-$93; $E4-$E7	ACTIVE DISPLAY: cursor is visible & moving
$E8-$FF; $00-$17	INACTIVE

This game does latch HVC but adds a default X offset of **$18**. Note that the counter gap is correctly taken in account by the software routine.

Y Position

No offset is applied by default, the V counter value is directly converted into Y coordinate, active range being $00-$DF.