

# open\_game\_module.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/04/19

---

## INFORMATION

---

### Brief

---

Colecovision Emulation of the Super Game Module using a YMZ284 for audio.

### License MIT

---

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## open\_game\_module

---

```
module open_game_module (  
    input  
    clk,  
  
    10:0]  
    A,  
    input  
    MREQn,  
    input  
    RFSHn,  
    input  
    IORQn,  
    input  
    WRn,
```

input [

```
input
RESETn,
input
RDn,

7:0]
D,
output
RAM_CSn,
output
RAM_OEn,
output
AY_CSn,
output
AY_AS
)
```

inout [

Colecovision Super Game Module Glue Logic

Ports

clk	Clock for all devices in the core
input	
A	Address input bus from Z80
input[ 10: 0]	
MREQn	Z80 memory request input, active low
input	
RFSHn	Refresh request, active low
input	
IORQn	Z80 IO request input, active low
input	
WRn	Z80 Write to bus, active low
input	
RESETn	Input for reset, active low
input	
RDn	Z80 Read from bus, active low
input	
D	Z80 8 bit data bus, tristate IN/OUT
inout[ 7: 0]	
RAM_CSn	RAM chip select, active low
output	
RAM_OEn	RAM Ouput enable, active low
output	
RAM_WEn	RAM write enable, active low
AY_CSn	AY sound chip chip select
output	
AY_AS	AY data or register select
output	

REGISTER INFORMATION

Core has 3 registers at the IO addresses that follow.

SOUND_ADDR_CACHE	h50
SOUND_CACHE	h51
RAM_24K_ENABLE	h53

**SWAP\_BIOS\_TO\_RAM** h7F

## SOUND\_ADDR\_CACHE

```
localparam SOUND_ADDR_CACHE = 8'h50
```

Defines the address of r\_snd\_addr\_cache

SOUND ADDR CACHE REGISTER	
1:0	
CACHE LAST ADDRESS WRITE TO AY SOUND CHIP, BITS 2:1	

Setup an address for cache the sound address so each write will be to a proper address (opcode games need to write multiple and read multiple addresses) The register is only 4 bits since there are only 16 registers max.

## SOUND\_CACHE

```
localparam SOUND_CACHE = 8'h51
```

Defines the address of r\_snd\_cache

SOUND CACHE REGISTER	
7:0	
CACHE LAST WRITE TO AY SOUND CHIP	

Cache Sound Chip as the SGM games read from it (Yamaha chip does not have a read like a GI does).

## RAM\_24K\_ENABLE

```
localparam RAM_24K_ENABLE = 8'h53
```

Defines the address of r\_24k\_ena

24K RAM ENABLE REGISTER	
7:1	0
ZERO	ENABLE 24K RAM, ACTIVE HIGH

Super Game Module 24K RAM enable using bit 0 (Active High)

## SWAP\_BIOS\_TO\_RAM

```
localparam SWAP_BIOS_TO_RAM = 8'h7F
```

Defines the address of r\_swap\_ena

SWAP BIOS TO RAM REGISTER			
7:4	3:2	1	0
ZERO	ONE	BIO TO RAM SWAP, ACTIVE LOW	ONE

Super Game Module BIOS to RAM swap on bit 1 (Active Low)

## r\_snd\_addr\_cache

```
reg [ 1:0] r_snd_addr_cache = 0
```

register for SOUND\_ADDR\_CACHE See Also: [SOUND\\_ADDR\\_CACHE](#)

## r\_24k\_ena

```
reg [ 7:0] r_24k_ena = 0
```

register for RAM\_24K\_ENABLE See Also: [RAM\\_24K\\_ENABLE](#)

## r\_swap\_ena

```
reg [ 7:0] r_swap_ena = 8'h0F
```

register for 8K RAM/ROM swap See Also: [SWAP\\_BIOS\\_TO\\_RAM](#)

## r\_snd\_cache

```
reg [ 7:0] r_snd_cache[3:0]
```

register for SOUND\_CACHE See Also: [SOUND\\_CACHE](#)

## ASSIGNMENT INFORMATION

How signals are created

## s\_enable

```
assign s_enable = (  
  RFSHn &  
  
  MREQn  
)
```

Decided to keep the same method used internally as the coleco. This emualtes the original ttl chip logic.

## s\_y0\_selIn

```

assign s_y0_sel0 = ~(
                                s_enable & ~A[10] & ~
A[9] &
A[8]
)

```

Address h0000, ROM/RAM

**s\_enable**     Enable decoder

**A[10:8]**     Address lines used for select lines (actually lines A[15:13]).

---

## s\_ram2\_csn

```

assign s_ram2_csn = ~(
                                s_enable & ~A[10] & ~
A[9] &
A[8]
)

```

Address h2000, RAM

**s\_enable**     Enable decoder

**A[10:8]**     Address lines used for select lines (actually lines A[15:13]).

---

## s\_ram1\_csn

```

assign s_ram1_csn = ~(
                                s_enable & ~A[10] & ~
A[9] &
A[8]
)

```

Address h4000, RAM

**s\_enable**     Enable decoder

**A[10:8]**     Address lines used for select lines (actually lines A[15:13]).

---

## s\_ram0\_csn

```

assign s_ram0_csn = ~(
                                s_enable & ~A[10] & ~
A[9] &
A[8]
)

```

Address h6000, RAM

**s\_enable**     Enable decoder

**A[10:8]**     Address lines used for select lines (actually lines A[15:13]).

---

## s\_ram\_csn

```
assign s_ram_csn = (
  (s_y0_seln | r_swap_ena[1]) & (s_ram2_csn | ~r_24k_ena[0]) & (s_ram1_csn |
    s_ram0_csn | ~r_24k_ena[0])
)
```

RAM Chip select when address is requested (active low). When the 24k is not enabled, use internal memory.

<b>(s_y0_seln   r_swap_ena[1])</b>	address range starting at h0000, swap bios/rom bit is enabled (1 is disabled).
<b>(s_ram1_csn   ~r_24k_ena[0])</b>	address range starting at h4000, 24k enable bit from register.
<b>(s_ram2_csn   ~r_24k_ena[0])</b>	address range starting at h2000, 24k enable bit from register.
<b>(s_ram0_csn   ~r_24k_ena[0])</b>	address range starting at h6000, 24k enable bit from register.

## RAM\_OEn

```
assign RAM_OEn = RDn | s_ram_csn
```

RAM Output enable when read is requested (active low).

**RDn** Z80 read request, active low.

**s\_ram\_csn** See Also: [s\\_ram\\_csn](#)

## RAM\_CSn

```
assign RAM_CSn = s_ram_csn
```

RAM Chip Select output assignment.

**s\_ram\_csn** See Also: [s\\_ram\\_csn](#)

## DECODER INFORMATION FOR SUPER GAME MODULE

How address decoder is created for Super Game Module, using a YMZ284.

**SGM IO REG** Clocked IO decoder for Super Game Module.

## AY\_AS

```
assign AY_AS = (
  A[7:0]
  =
  = 8'h50 & ~IORQn & ~WRn ? 1'b0 : 1'b1
)
```

h50 is the address select, when selected its in data mode

**A[7:0]** If address matches h50, enable

**IORQn** Active IO request, enable

**WRn** Z80 write is active, enable

## s\_ay\_sound\_csn

---

match both h50 and h51 by ignoring bit 0. Enable AY sound chip.

**A[7:0]**      If address matches h50 or h51, enable

**IORQn**      Active IO request, enable

**WRn**        Z80 write is active, enable

## D

---

```
assign D = (
  A[7:0]
  =
  = 8'h52 & ~IORQn & ~RDn ? r_snd_cache[{2'b00, r_snd_addr_cache}] : 8'bzzzzz
)
```

read cached register from previous write (AY emulation), at set address location.

**A[7:0]**      If address matches h52, enable

**IORQn**      Active IO request, enable

**RDn**        Z80 read is active, enable