

# porta\_glue\_coleco.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/11/06

---

## INFORMATION

---

### Brief

---

Colecovision SGM glue logic chip

### License MIT

---

Copyright 2024 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.

## CONSTANTS

---

### DEF\_RESET\_DELAY\_BIT

---

Number of bits for reset delay register

### DEF\_FB\_MONOSTABLE\_COUNT

---

delay till state is at 1 instead of 0 (its stable state) for feedback stable circuit

### DEF\_IRQ\_MONOSTABLE\_COUNT

---

delay till state is at 1 instead of 0 (its stable state) for the controller (spinner) generated interrupt.

## porta\_glue\_coleco

---

```
module porta_glue_coleco (  
  input  
  clk,  
  
  15:0]  
  A,  
  input  
  C1P1,  
  input  
  C1P2,  
  input  
  C1P3,  
  input  
  C1P4,  
  input  
  C1P6,  
  input  
  C1P7,  
  input  
  C1P9,  
  input  
  C2P1,  
  input  
  C2P2,  
  input  
  C2P3,  
  input  
  C2P4,  
  input  
  C2P6,  
  input  
  C2P7,  
  input  
  C2P9,  
  input  
  MREQn,  
  input  
  IORQn,  
  input  
  RFSHn,  
  input  
  M1n,  
  input  
  WRn,  
  input  
  RESETn_SW,  
  input  
  RDn,  
  
  7:0]  
  D,  
  output  
  CP5_ARM,  
  output  
  CP8_FIRE,  
  output  
  CS_h8000n,  
  output  
  CS_hA000n,  
  output
```

input [

inout [

```

CS_hC000n,
output
CS_hE000n,
output
SND_ENABLEn,
output
ROM_ENABLEn,
output
RAM_CSn,
output
RAM_OEn,
output
CSWn,
output
CSRn,
output
WAITn,
output
RESETn,
output
RAM_MIRRORn,
output
INTn,
output
AS,
output
AY_SND_ENABLEn
)

```

Colecovision Super Game Module Glue Logic

## Ports

<b>clk</b> input	Clock for all devices in the core
<b>A</b> input[ 15: 0]	Address input bus from Z80
<b>C1P1</b> input	DB9 Controller 1 Pin 1
<b>C1P2</b> input	DB9 Controller 1 Pin 2
<b>C1P3</b> input	DB9 Controller 1 Pin 3
<b>C1P4</b> input	DB9 Controller 1 Pin 4
<b>C1P6</b> input	DB9 Controller 1 Pin 6
<b>C1P7</b> input	DB9 Controller 1 Pin 7
<b>C1P9</b> input	DB9 Controller 1 Pin 9
<b>C2P1</b> input	DB9 Controller 2 Pin 1
<b>C2P2</b> input	DB9 Controller 2 Pin 2
<b>C2P3</b> input	DB9 Controller 2 Pin 3
<b>C2P4</b> input	DB9 Controller 2 Pin 4
<b>C2P6</b>	DB9 Controller 2 Pin 6

input	
<b>C2P7</b> input	DB9 Controller 2 Pin 7
<b>C2P9</b> input	DB9 Controller 2 Pin 9
<b>MREQn</b> input	Z80 memory request input, active low
<b>IORQn</b> input	Z80 IO request input, active low
<b>RFSHn</b> input	Z80 Refresh input, active low
<b>M1n</b> input	Z80 M1 state, active low
<b>WRn</b> input	Z80 Write to bus, active low
<b>RESETn_SW</b> input	Input for reset switch
<b>RDn</b> input	Z80 Read from bus, active low
<b>D</b> inout[ 7: 0]	Z80 8 bit data bus, tristate IN/OUT
<b>CP5_ARM</b> output	DB9 Controller 1&2 ARM Select
<b>CP8_FIRE</b> output	DB9 Controller 1&2 FIRE Select
<b>CS_h8000n</b> output	Select when Z80 requests memory at h8000 (GAME CART), active low
<b>CS_hA000n</b> output	Select when Z80 requests memory at hA000 (GAME CART), active low
<b>CS_hC000n</b> output	Select when Z80 requests memory at hC000 (GAME CART), active low
<b>CS_hE000n</b> output	Select when Z80 requests memory at hE000 (GAME CART), active low
<b>SND_ENABLEn</b> output	SN76489 Sound chip enable, active low
<b>ROM_ENABLEn</b> output	Enable BIOS ROM, active low
<b>RAM_CSn</b> output	RAM chip select, active low
<b>RAM_OEn</b> output	RAM Ouput enable, active low
<b>CSWn</b> output	Chip Select Write for VDP, active low
<b>CSRn</b> output	Chip Select Read for VDP, active low
<b>WAITn</b> output	Wait state generator for Z80, active low
<b>RESETn</b> output	Timed reset generated by Logic, active low
<b>RAM_MIRRORn</b> output	Extended RAM, high is extended RAM, active low is mirrored.
<b>INTn</b> output	Interrupt generator for Z80, active low

**AS**                                      AY sound chip address(0)/data(1) select  
**output**

**AY\_SND\_ENABLEn**                      AY sound enable, active low  
**output**

## REGISTER INFORMATION

Core has 3 registers at the addresses that follow.

**SOUND\_CACHE**                      h51  
**RAM\_24K\_ENABLE**                      h53  
**SWAP\_BIOS\_TO\_RAM**                      h7F

### 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\_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 = 0
```

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

## r\_int\_p1

```
reg r_int_p1 = 1'b0
```

Interrupt from player one control

## r\_int\_p2

```
reg r_int_p2 = 1'b0
```

Interrupt from player two control

## r\_wait

```
reg r_wait = 1'b0
```

Wait state generated register

## r\_reset\_counter

```
reg [ 9:0] r_reset_counter = 0
```

---

Timed reset counter

## **r\_resetn**

---

```
reg r_resetn = 0
```

Registered reset output, active low

## **r\_mono\_count\_p1**

---

```
reg [11:0] r_mono_count_p1 = 0
```

monostable circuit counters, player 1 AND

## **r\_mono\_count\_p2**

---

```
reg [11:0] r_mono_count_p2 = 0
```

monostable circuit counters, player 2 AND

## **r\_mono\_count\_int\_p1**

---

```
reg [ 5:0] r_mono_count_int_p1 = 0
```

monostable circuit counters, player 1 interrupt

## **r\_mono\_count\_int\_p2**

---

```
reg [ 5:0] r_mono_count_int_p2 = 0
```

monostable circuit counters, player 2 interrupt

## **r\_mono\_p1**

---

```
reg r_mono_p1 = 1'b0
```

Feedback from IRQ to controller 1 register

## **r\_mono\_p2**

---

```
reg r_mono_p2 = 1'b0
```

Feedback from IRQ to controller 2 register

## **r\_ctrl\_fire**

---

```
reg r_ctrl_fire = 1'b1
```

NAND Feedback Flip Flop FIRE select.

## **r\_ctrl\_arm**

---

```
reg r_ctrl_arm = 1'b0
```

NAND Feedback Flip Flop ARM select.