Mini Colecovision

November 6, 2024

Jay Convertino

Contents

1 Usage

1.1 Introduction

This manual describes how to use the RODAC (Retro Only Device Application Creation) system for development. Items such as how to build included apps, what the structure of the system looks like, and how to create your own app is included. The final section are links to doxygen generatated documentation about the drivers used by this system.

1.2 Dependecies

The following are the dependecies needed to build the applications targeting various retro systems.

- sdcc 4.X.X
- python 3.X
- make

1.2.1 Protable Coleco Glue File List

- src
  - 'src/porta_glue_coleco.v': 'file_type': 'verilogSource'
- constr
  - 'constr/porta_glue_coleco.sdc': 'file_type': 'SDC'
- tb
  - 'tb/tb_porta_glue_coleco.v': 'file_type': 'verilogSource'

1.2.2 Protable Coleco Glue Targets

- default

  Info: Default IP target for future tool intergration.
  - src
  - constr
- sim

  Info: Simulation target for basic test bench.
  - src
  - tb

1.3 Building

Makefiles are used to execute all builds. All sources will rebuild when make is run due to the ability to change the system target. If this didn't happen the new memory map setup in the defines.h would not be applied. Each application has its makefile located in its root folder. To run a build you must run, in the target apps root folder, the following command.

```
$ make SYSTEM
```

Where system is the target you would like to build for. All will do nothing but through an error telling you the same. Currently the targets are **coleco, coleco_sgm, msx, sg1000**. All targets have been tested for coleco based systems. The others are not tested on real hardware at the moment.

1.3.1 hello_world

Hello World is a simple application that prints all of the characters from the TMS memory to screen. It also prints hello world in the center of the screen and scrolls it. This is done in the TMS txt mode with 40 columns and no sprites. This application has been tested in emulation on all available systems. It also generates a single annoying constant tone, as a really poor sound test. To build this run the following for the Colecovision in the root of the apps/hello_world folder.

```
$ make coleco
```

1.3.2 multicart

Multicart creates a non-scrolling list of ROMs in alphabetical order. The number of ROMs is limited by the target flash size and the number of lines on screen (till scrolling is added, currently 21). The generation of the header that contains the list of ROMs is automatic with a python script, rom_header_gen.py. The full ROM is also auto generated by a python script rom_file_gen.py. Currently these default to the roms folder located in the apps/multicart folder root. This can be changed in the make file via the ROM_ variables. There are dummy ROMs with random data for testing of the system. This will work in an emulator up to the point of bank switching ROMs, since the PIC and the logic with it is not emulated. This is currently only targeted and tested on the Colecovision. To build for the Colecovision you would run the following in the apps/multicart folder.

```
$ make coleco
```

1.4 Directory Guide

Below highlights important folders from the root of RODAC.

1. **docs** Contains all documentation related to this project.
    - **arch** Contains all architecture docs related to retro systems.
    - **manual** Contains user manual and wiki that are generated from the same latex source.
2. **apps** Contains source code in C for the applications to run on the target architecture.
    - **hello_world** Example hello world application. Targets all architectures.
    - **mutlicart** Example multicart application, written for the coleco only.
3. **drivers** Contains all source code related to the project.
    - **gisnd** Simple driver for the GI AY-3-8910 sound chip and its variants.
    - **sn76489** driver for the TI SN76489 sound chip.
    - **tms99XX** driver for all TMS99XX and TMS9XXX video chips.

2 Application Creation

3 System Creation

4 Module Documentation

porta_glue_coleco.v

AUTHORS

JAY CONVERTINO

DATES

2024/11/06

INFORMATION

Brief

Colecovision SGM glue logic chip

License MIT

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,
                                                                input [
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,
                                                                inout [
7:0]
D,
output
CP5_ARM,
output
CP8_FIRE,
output
CS_h8000n,
output
CS_hA000n,
output
```

```
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

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

input

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

AS output	AY sound chip address(0)/data(1) select
AY_SND_ENABLEn output	AY sound enable, active low

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.