MAZINGER Z

SEGA SATURN

DEBUG REFERENCE

SUMMARY

このドキュメントの目的は、Jo-Engineを使用してビデオゲームをデバッグする簡単な方法を実用的な方法で説明することです。

(c) Rolando Fernández Benavidez 2019.

MAZINGER Z HOMEBREW FOR SEGA SATURN

使用許諾

This program is distributed in the hope that it will be useful, but without any warranty, without even the implied warranty of particular purpose.

Nova32 Development Software
(c) Rolando Fernández Benavidez, March 2019.

M A Z I N G E R Z -HOMEBREW-

マジンガーz
セガサターンで動作するようにプログラムされたアクションゲーム。

use only
GNU GCC-COFF & Jo-Engine

このプログラムはフリーソフトウェアです：あなたはそれを再配布することができますおよび/
または変更することができます
によって公開されているGNU一般公衆利用許諾契約書の条項の下で

Free Software Foundation、ライセンスのバージョン3、または
（あなたの選択で）それ以降のバージョン。

このプログラムは、役に立つことを願って配布されています。
しかし、いかなる保証もありません。 の黙示的な保証もありません。

商品性または特定の目的への適合性。 を参照してください
詳細については、GNU General Public License。

あなたはGNU General Public Licenseのコピーを受け取っているはずです。
このプログラムと一緒に。　そうでない場合は、<https://www.nova32.com.mx>を参照してください。<（c）RolandoFernándezBenavidez 2019>。

Platform: SEGA SATURN
Version 3.0.0

NOVA32 DEVELOPMENT
SOFTWARE

CONTENT OF THE MANUAL

マニュアルの内容

NOVA32 DEVELOPMENT
SOFTWARE

I. CODE EDITION

You can use any simple text editor, or some environment to develop software, in that case it will only allow you to edit your code; mainly because the compilation must be done with the kit provided by Jo-Engine, on the Johannes Fetz website. Author of the Jo-Engine.

In my case, I like to use Notepad ++, it's pretty good for any kind of project; It is definitely my favorite editor, but you can take another one.

You can edit the source code:

```
/*****************************************************************************
*                                                                            *
*    This program is distributed in the hope that it will be useful,         *
*    but without any warranty, without even the implied warranty of          *
*    particular purpose.                                                     *
*                                                                            *
*     Nova32 Development Software                                            *
*     (c) Rolando Fernandez Benavidez, March 2019.                           *
*                                                                            *
*****************************************************************************
*                        M A Z I N G E R   Z   -HOMEBREW-                    *
*                                                                            *
*                                  マジンガーz                               *
*                                                                            *
*                    セガサターンで動作するようにプログラムされたアクションゲーム。          *
*                                                                            *
*                                  use only                                  *
*                              GNU GCC-COFF & Jo-Engine                      *
*                                                                            *
*    このプログラムはフリーソフトウェアです:あなたはそれを再配布することができますおよび/      *
*    または変更することができます                                              *
*    によって公開されているGNU一般公衆利用許諾契約書の条項の下で                *
*    Free Software Foundation、ライセンスのバージョン3、または                  *
*    (あなたの選択で)それ以降のバージョン。                                     *
*                                                                            *
*    このプログラムは、役に立つことを願って配布されています。                   *
*    しかし、いかなる保証もありません。 の黙示的な保証もありません。            *
*    商品性または特定の目的への適合性。 を参照してください                     *
*    詳細については、GNU General Public License。                             *
*                                                                            *
*    あなたはGNU General Public Licenseのコピーを受け取っているはずです。      *
*    このプログラムと一緒に。 そうでない場合は、<https://www.nova32.com.mx>を参照してください。 *
*    <(c)RolandoFernándezBenavidez 2019>。                                   *
*                                                                            *
* Platform: SEGA SATURN                                                      *
*                                                                            *
* Version 3.0.0                                                              *
*****************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <sgl.h>
#include <jo/jo.h>
#include "global.h"
#include "mazinger.h"
```

```
25  *                                                                      *
26  *    このプログラムは、役に立つことを願って配布されています。          *
27  *    しかし、いかなる保証もありません。 の黙示的な保証もありません。   *
28  *    商品性または特定の目的への適合性。 を参照してください            *
29  *    詳細については、GNU General Public License。                      *
30  *                                                                      *
31  *    あなたはGNU General Public Licenseのコピーを受け取っているはずです。 *
32  *    このプログラムと一緒に。 そうでない場合は、<https://www.nova32.com.mx>を参照してください。 *
33  *    <(c)RolandoFernándezBenavidez 2019>。                             *
34  *                                                                      *
35  * Platform: SEGA SATURN                                                *
36  *                                                                      *
37  * Version 3.0.0                                                        *
38  ************************************************************************/
39
40  #ifndef __ASTEROID_H__
41  # define __ASTEROID_H__
42
43
44  #define MAZINGER_DEBUG
45
46  //Functions prototype:
47
48  //関数定義を含むヘッダファイル
49                                                  //この関数は日立SH-2 CPUのテストを実行することを可能にし、
50  void Exec_Assembler_Hitachi_SH2_CPU(void);      //SDKがSEGA Saturnハードウ
51  void OnWinner();                                //ェアへのフルアクセスでCOFFフォーマットでバイナリをコンパイルす
52  void SetBackground(void);                       //ることを可能にすることを実証します。
53  int  Collision(PLAYER *A,PLAYER *B);
54  void Stars(void);
55  void Frame(void);
56  void play_song(void);
57  void OnGameOver();
58  void Update();
59  void Awake();
60  void Start();
61  void Splash();
62  void Debug(void);
63  void PowerAnimation();
64  void BulletAnimation();
65  void FireAnimation();
66  void Init_mode7();
67  void Render_mode7();
68
69  #endif
70
```

```
454     //---------------------Saturn PAD button events------------------//
455
456     if (jo_is_pad1_key_pressed(JO_KEY_A))
457     {
458         jo_audio_play_sound_on_channel(&snd_bullet, 0);
459         FBullet.x = Player0.x + 14;
460         FBullet.y = Player0.y - 7;
461         Render_Bullet = TRUE;
462     } else Render_Bullet = FALSE;
463
464     if (jo_is_pad1_key_pressed(JO_KEY_C))
465     {
466         jo_audio_stop_cd();
467         jo_goto_boot_menu();
468     }
469
470     if (jo_is_pad1_key_pressed(JO_KEY_X))
471     {
472         jo_core_suspend(); //PAUSE, press [START] to continue...
473     }
474
```

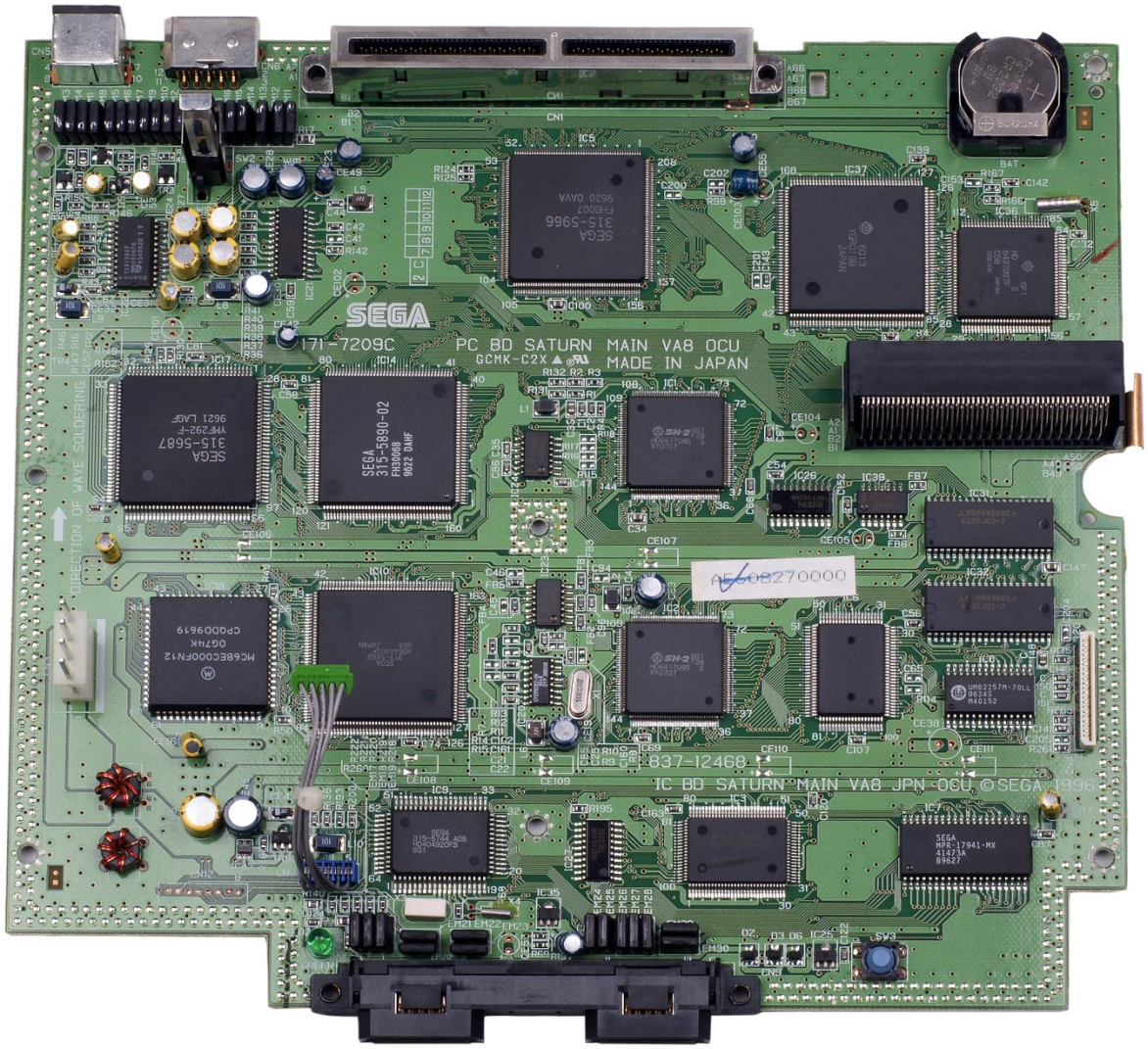
A simple function is included to show that it is possible to have full access to the SEGA Saturn hardware. In the example you can see a simple opcode execution TEST for the Hitachi SH-2 Master CPU of the Motherboard.

```
43    #include <math.h>
44    #include <sgl.h>
45    #include <jo/jo.h>
46    #include "global.h"
47    #include "mazinger.h"
48
49    ////////////////////////////////////////////////////////////////////////////
50    // Function: Exec_Assembler_Hitachi_SH2_CPU                               //
51    // Summary : HITACHI SH-2 MASTER CPU TEST                                 //
52    ////////////////////////////////////////////////////////////////////////////
53
54    void Exec_Assembler_Hitachi_SH2_CPU(void)
55    {
56
57
58    // この関数は日立SH-2 CPUのテストを実行することを可能にし、SDKがSEGA Saturnハードウ
59    // ェアへのフルアクセスでCOFFフォーマットでバイナリをコンパイルすることを可能にすることを実証します。
60
61        
62        asm ("      .ORG    H'1006          ");
63        asm ("      MOV     #H'7B1 , R1     ");  // STAGES=3 CYCLES=1
64        asm ("      MOV     #H'7C4 , R5     ");  // STAGES=3 CYCLES=1
65        asm ("      NOT        R5  , R1     ");
66        asm ("      ADD     #-1    , R0     ");  // STAGES=3 CYCLES=1
67        asm ("      TST     R0     , R0     ");  // STAGES=6 CYCLES=1
68        asm ("      SETT                    ");
69        asm ("      NEG     R1     , R0     ");
70        asm ("      .ALIGN  2               ");
71        asm ("      CLRMAC                  ");
72        asm ("      CLRT                    ");
73        asm ("DATAS1:                       ");
74        asm ("      MOV     #H'4D3 , R0     "); // STAGES=3 CYCLES=1
75        asm ("      MOV     #H'7C3 , R1     "); // STAGES=3 CYCLES=1
76        asm ("OPERATION:                    ");
77        asm ("      ADD        R0 , R1      "); // STAGES=3 CYCLES=1
78        asm ("      NOT        R0 , R1      ");
79        asm ("      STS       MACH , R1     "); // STAGES=4 CYCLES=1
80        asm ("      SUB        R0 , R1      ");
81        asm ("      MULU       R2 , R0      "); // STAGES=6/7*3 CYCLES=1
82        asm ("NOP                           ");
83        asm ("TARGET:                       ");
84        asm ("      .ALIGN  4               ");
85        asm ("      SETT                    ");
86        asm ("      OR         R5 , R0      ");
87        asm ("DATAS2:                       ");
88        asm ("      MOV     #H'00A , R0     "); // STAGES=3 CYCLES=1
```

In the example it is possible to see the stages and cycles consumed by the SH-2 CPU of the console.

The SEGA Saturn has two RISC processors, which work in a MASTER and SLAVE configuration:

SEGA Saturnには2つのRISCプロセッサがあり、それらはMASTERおよびSLAVE構成で動作します。



The Jo-Engine written by Johannes Fetz, allows to balance the workloads between one CPU and another, allowing to program an efficient game in the SEGA Saturn.

Johannes Fetzによって書かれたJo-Engineは、あるCPUと別のCPUの間でワークロードのバランスをとることを可能にし、SEGA Saturnで効率的なゲームをプログラムすることを可能にします。

ハードウェアの概要とハードウェアの基本的な説明

Processors

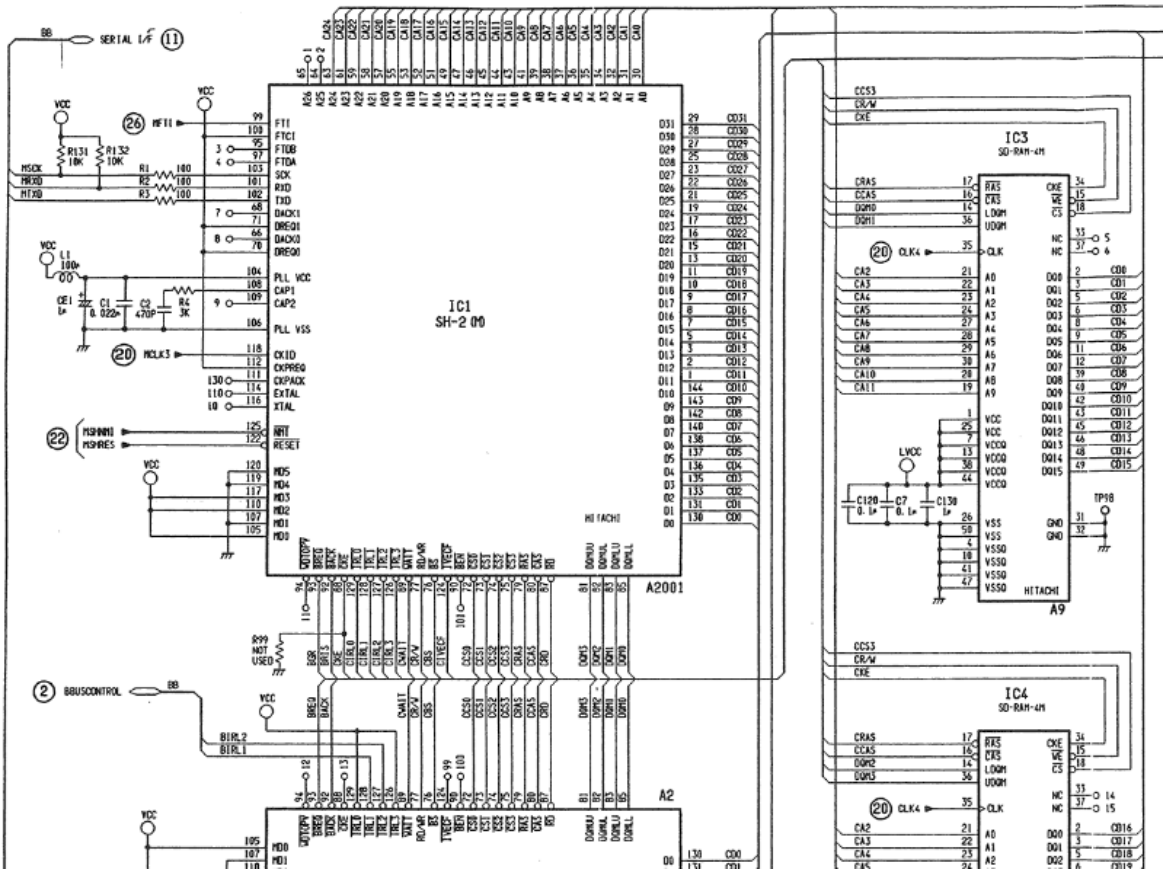
- Main CPU: 2× Hitachi SH-2 @ 28.63636 MHz
  - Configuration: Master/Slave
  - 2x CPU cores: 32-bit RISC instructions/registers, 74.454536 MIPS (37.227268 MIPS each, 1.3 MIPS per MHz), up to 4 instructions/cycle (2 instructions/cycle per SH-2)
  - 2x DMA units: 2x DMAC (Direct Memory Access Controller), parallel processing
  - 4x internal fixed-point math processors: 2x MULT multiplier DSP, 2x DIVU division units, parallel processing
  - 2x MULT multiplier DSP: 57.27272 MOPS fixed-point math (28.63636 MOPS per SH-2)
  - 2x DIVU division units: 16/32/64-bit division, 1,468,531 divides/sec
  - Bus width: 64-bit (2× 32-bit) internal, 32-bit external
  - Word length: 32-bit
- System coprocessor: Custom Saturn Control Unit (SCU), with DSP for geometry processing and DMA controller for system control
  - System control processor: 32-bit fixed-point registers/instructions, interrupt controller, DMA controller, 3 DMA channels
  - Math coprocessor: Geometry DSP @ 14.31818 MHz, 32-bit fixed-point instructions, 6 parallel instructions per cycle, 85.90908 MIPS (6 MIPS per MHz)
- CD-ROM CPU: Hitachi SH-1 32-bit RISC processor @ 20 MHz (20 MIPS) (controlling the CD-ROM)
  - Contains internal DAC and internal math processor
  - Bus width: 32-bit internal, 16-bit external
- Microcontroller: Hitachi HD404920 (4-bit MCU) "System Manager & Peripheral Control" (SMPC) @ 4 MHz
  - RTC: 1 MHz (real-time clock)
  - Instruction set: 4-bit instructions, 890 ns per instruction, 1.123595 MIPS
  - Bus width: 10-bit internal, 8-bit external
- Optional MPEG Video CD Card:
  - MPEG Video decoder: Sega P/N 315-5765 (Hitachi HD814101FE)
  - MPEG Audio decoder: Hitachi HD814102F

Audio

- Sound processor: Yamaha SCSP (Saturn Custom Sound Processor) YMF292
  - Sound DSP: Yamaha FH1 DSP (Digital Signal Processor) @ 22.58 MHz (24-bit, 128-step, 4 parallel instructions)
  - Bus width: 24-bit internal, 16-bit external
- Sound CPU: Motorola 68EC000 (16/32-bit CISC) sound processor @ 11.29 MHz (1.97575 MIPS)
  - Bus width: 16-bit internal, 16-bit external

Video

- Sega/Hitachi VDP1 @ 28.63636 MHz: Handles sprite/texture and polygon drawing
    - Bus width: 48-bit (3x 16-bit)
    - Word length: 16-bit
- Sega/Yamaha VDP2 @ 28.63636 MHz: Backgrounds, scrolling, handles background, scroll and 3D rotation planes
    - Bus width: 32-bit
    - Word length: 32-bit
- Sony CXA1645M RGB-Composite Video Encoder



II. VIDEO GAME ARCHITECTURE

MAZINGER Z HOMEBREW

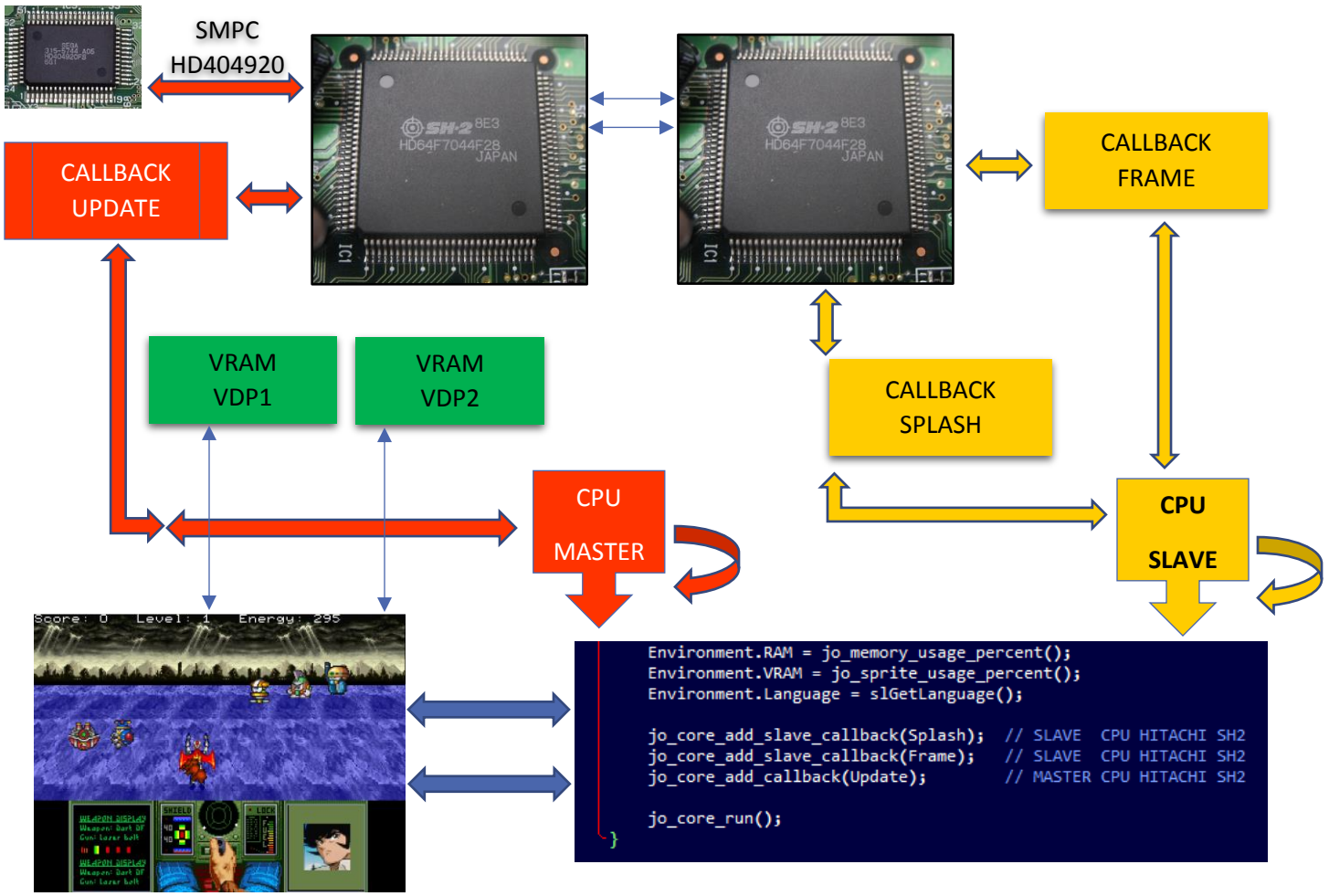
マジンガーz

セガサターンで動作するようにプログラムされたアクションゲーム。

This video game has as its fundamental objective, to show in a practical way an example of a game in the SEGA Saturn hardware. It can be used as a reference material in any kind of videogame programming with C language.

Mazinger Z Homebrew, was programmed using the Jo-Engine. Attempts are made to use the Master-Slave CPU characteristics, for efficient code performance.

The architecture of the game Mazinger Z Homebrew, マ ジ ン ガ ー z can be seen in the following conceptual design:



III. COMPILATION

As mentioned in the previous lines, the compilation must be done with the development kit provided with Jo-Engine, on the Johannes Fetz website.

Example of compilation output with the MAKE command:

```
255 -DJO_MAX_FILE_IN_IMAGE_PACK=32 -DJO_MAP_MAX_LAYER=8 -DJO_MAX
-DJO_MAX_FS_BACKGROUND_JOBS=4 -DJO_DEBUG -DJO_NTSC_VERSION -DJO_
_CARD_SUPPORT -DJO_COMPILE_WITH_DUAL_CPU_SUPPORT -fkeep-inline-f
l -Wshadow -Wbad-function-cast -Winline -Wcomment -Winline -Wlon
mpare -Wextra -Wno-strict-aliasing --param max-inline-insns-sing
sions -std=gnu99 -fmerge-all-constants -fno-ident -fno-unwind-ta
ronous-unwind-tables -fomit-frame-pointer -fstrength-reduce -fre
 -nodefaultlibs -nostdlib -fno-builtin -m2 -c -I../../jo_engine
/SGL_302j/INC  -o ../../jo_engine/vdp1_command_pipeline.o
../../Compiler/SH_COFF/sh-coff/bin/sh-coff-gcc.exe -m2 -L../../C
/LIB_COFF -Xlinker --format=coff-sh -Xlinker -T../../Compiler/CO
-Xlinker -Map -Xlinker sl_coff.map -Xlinker -e -Xlinker ___Start
../../Compiler/SGL_302j/LIB_COFF/SGLAREA.O main.o mazinger.o ../
.o ../../jo_engine/audio.o ../../jo_engine/fs.o ../../jo_engine/
_engine/font.o ../../jo_engine/input.o ../../jo_engine/physics.o
/core.o ../../jo_engine/math.o ../../jo_engine/malloc.o ../../jo
../../jo_engine/sprites.o ../../jo_engine/map.o ../../jo_engine/
engine/sprite_animator.o ../../jo_engine/image.o ../../jo_engine
/../jo_engine/time.o ../../jo_engine/vdp1_command_pipeline.o ../
302j/LIB_COFF/SEGA_SYS.A ../../Compiler/SGL_302j/LIB_COFF/LIBCD.
/SGL_302j/LIB_COFF/LIBSGL.A -o sl_coff.coff
../../Compiler/SH_COFF/sh-coff/bin/sh-coff-objcopy.exe -O binary
_coff.bin
../../Compiler/SH_COFF/sh-coff/bin/sh-coff-objcopy.exe -O binary
/0.bin
mkisofs -quiet -sysid "SEGA SATURN" -volid "SaturnApp" -volset "
isher "SEGA ENTERPRISES, LTD." -preparer "SEGA ENTERPRISES, LTD.
App" -abstract "./cd/ABS.TXT" -copyright "./cd/CPY.TXT" -biblio
generic-boot ../../Compiler/COMMON/IP.BIN -full-iso9660-filename
 ./cd
------------------------------------------------------------
COMPILATION COFF FINISHED!
------------------------------------------------------------
RC= 0
Presione una tecla para continuar . . .
```

The configuration of the makefile file should be as follows:

```
JO_COMPILE_WITH_VIDEO_MODULE = 0
JO_COMPILE_WITH_BACKUP_MODULE = 0
JO_COMPILE_WITH_TGA_MODULE = 1
JO_COMPILE_WITH_AUDIO_MODULE = 1
JO_COMPILE_WITH_3D_MODULE = 0
JO_COMPILE_WITH_PSEUDO_MODE7_MODULE = 1
JO_COMPILE_WITH_EFFECTS_MODULE = 0
JO_DEBUG = 1
JO_COMPILE_USING_SGL=1
JO_COMPILE_WITH_DUAL_CPU_MODULE = 1
SRCS=main.c mazinger.c
JO_ENGINE_SRC_DIR=../../jo_engine
COMPILER_DIR=../../Compiler
include $(COMPILER_DIR)/COMMON/jo_engine_makefile

```

Clean object files before compiling:

```
C:\Saturn_SDK\Projects\MAZINGER>SET COMPILER_DIR=..\..\Compiler

C:\Saturn_SDK\Projects\MAZINGER>SET JO_ENGINE_SRC_DIR=../../jo_e

C:\Saturn_SDK\Projects\MAZINGER>SET PATH=..\..\Compiler\SH_COFF\
C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windo
stem32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;c:\Progr
icrosoft SQL Server\110\Tools\Binn\;c:\Program Files\Microsoft S
ols\Binn\;c:\Program Files\Microsoft SQL Server\110\DTS\Binn\;C:
x86)\Windows Kits\8.1\Windows Performance Toolkit\;C:\3dsmax5\ba
ogram Files (x86)\Common Files\Autodesk Shared\;C:\Program Files
ive\Shared;C:\Users\rolando\batariBasic;

C:\Saturn_SDK\Projects\MAZINGER>rm -f ./cd/0.bin

C:\Saturn_SDK\Projects\MAZINGER>rm -f *.o

C:\Saturn_SDK\Projects\MAZINGER>rm -f ../../jo_engine/*.o

C:\Saturn_SDK\Projects\MAZINGER>rm -f ./sl_coff.bin

C:\Saturn_SDK\Projects\MAZINGER>rm -f ./sl_coff.coff

C:\Saturn_SDK\Projects\MAZINGER>rm -f ./sl_coff.map

C:\Saturn_SDK\Projects\MAZINGER>rm -f ./sl_coff.iso

C:\Saturn_SDK\Projects\MAZINGER>rm -f ./sl_coff.cue
```

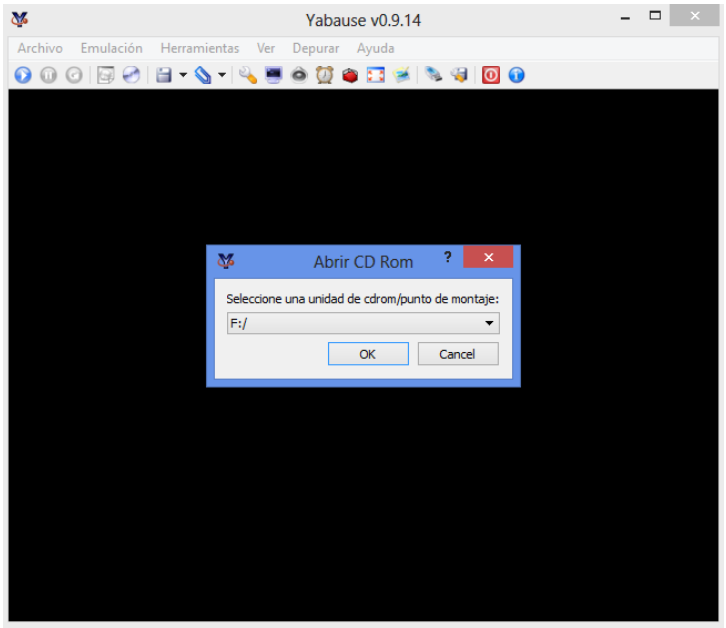
IV. EXECUTION OF THE VIDEO GAME

The resulting video game can be executed either in the real console or in an emulator. Mazinger Z Homebrew was tested on a SEGA Saturn North American model HST-3220. And developed using the Yabause emulator.

To run it in a real console, you must modify it to make use of the SWAP technique, you must have an original SEGA Saturn disk on hand, to start the system and quickly replace the disk in which the Mazinger Z Homebrew is located. It is recommended to create the CD-R at a speed of 4X so that the console can read it without problems.

Warning: Remember that the SWAP technique can damage the laser in your console!

If you do not have a SEGA Saturn console modified for SWAP, you can try the Mazinger Z Homebrew, using the Yabause emulator, in your preferred operating system.



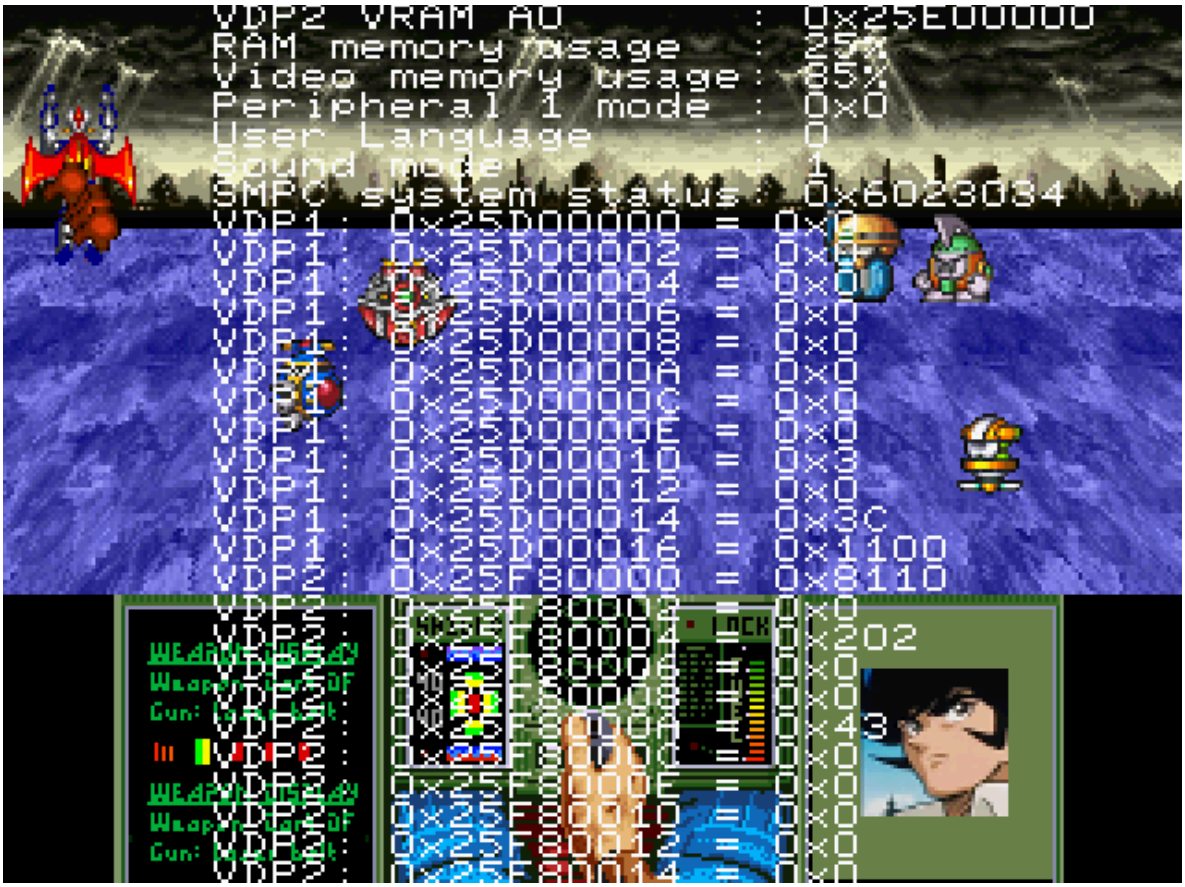
V. DEPURATION OF THE PROGRAM

Mazinger Z for the Saturn console, was developed with Homebrew software, as it does not have the official SEGA SDK, we do not have the professional tools to debug the programming code; therefore, we will have to make use of the classic outputs on the screen, to see any value of variables or the CPU registers and the VDP1 and VDP2 graphics chips.

Within the source code you will find an example function to be able to do debugging, the function has the following prototype: ***void Debug(void)***

```
179  ////////////////////////////////////////////////////////////////////////////
180  // Function: Debug                                                        //
181  // Summary : Debugger game                                                //
182  ////////////////////////////////////////////////////////////////////////////
183  void Debug(void)
184  {
185
186      if (jo_is_pad1_key_pressed(JO_KEY_B))
187      {
188          unsigned short  *ptr;
189          int l=7;
190
191          Debug_Show = TRUE;
192
193          jo_printf(7,0, "VDP2 VRAM A0      : 0x%X  ", VDP2_VRAM_A0);
194          jo_printf(7,1, "RAM memory usage  : %d%%  ", Environment.RAM);
195          jo_printf(7,2, "Video memory usage: %d%%  ", Environment.VRAM);
196          jo_printf(7,3, "Peripheral 1 mode : 0x%X  ", slGetPortMode1());
197          jo_printf(7,4, "User Language     : %d    ", slGetLanguage());
198          jo_printf(7,5, "Sound mode        : %d    ", slGetSoundOutput());
199          jo_printf(7,6, "SMPC system status: 0x%X  ", Smpc_Status); //Microcontroller Hitachi HD40492
200
201          for (ptr = (unsigned short *)JO_VDP1_REG; ptr <= (unsigned short *)JO_VDP1_LAST_REG; ++ptr)
202              {
203                 jo_printf(7,l, "VDP1: 0x%X = 0x%X", (int)ptr, *ptr);
204                 l++;
205              }
206
207          for (ptr = (unsigned short *)JO_VDP2_REG; ptr <= (unsigned short *)JO_VDP2_LAST_REG; ++ptr)
208              {
209                 jo_printf(7,l, "VDP2: 0x%X = 0x%X", (int)ptr, *ptr);
210                 l++;
211              }
212
213
214      }
215      else Debug_Show = FALSE;
216
217  }
```

The output of the COFF program in the running SEGA Saturn should look like this:



Is this the only way to debug in a SEGA Saturn?

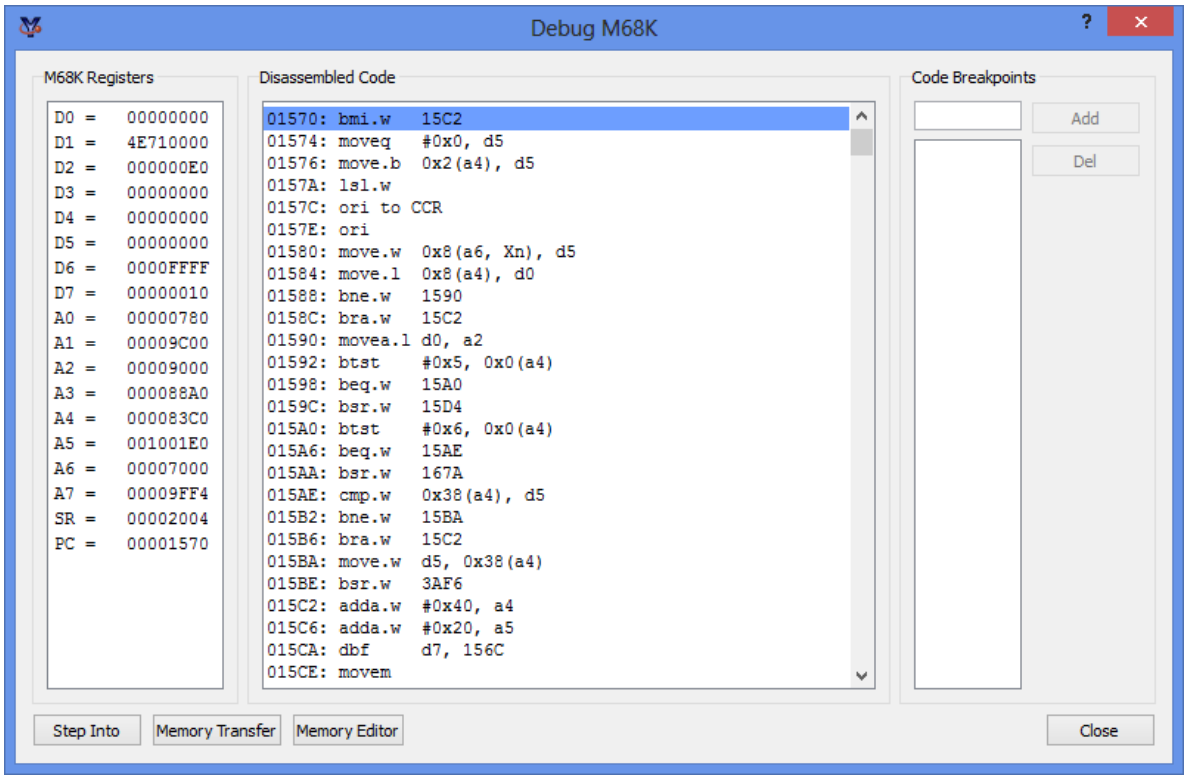
Surely you will be asking this question, do not worry, luckily the designers of the Yabause emulator, added a set of tools to be able to perform professional debugging or as close to the official SDK.

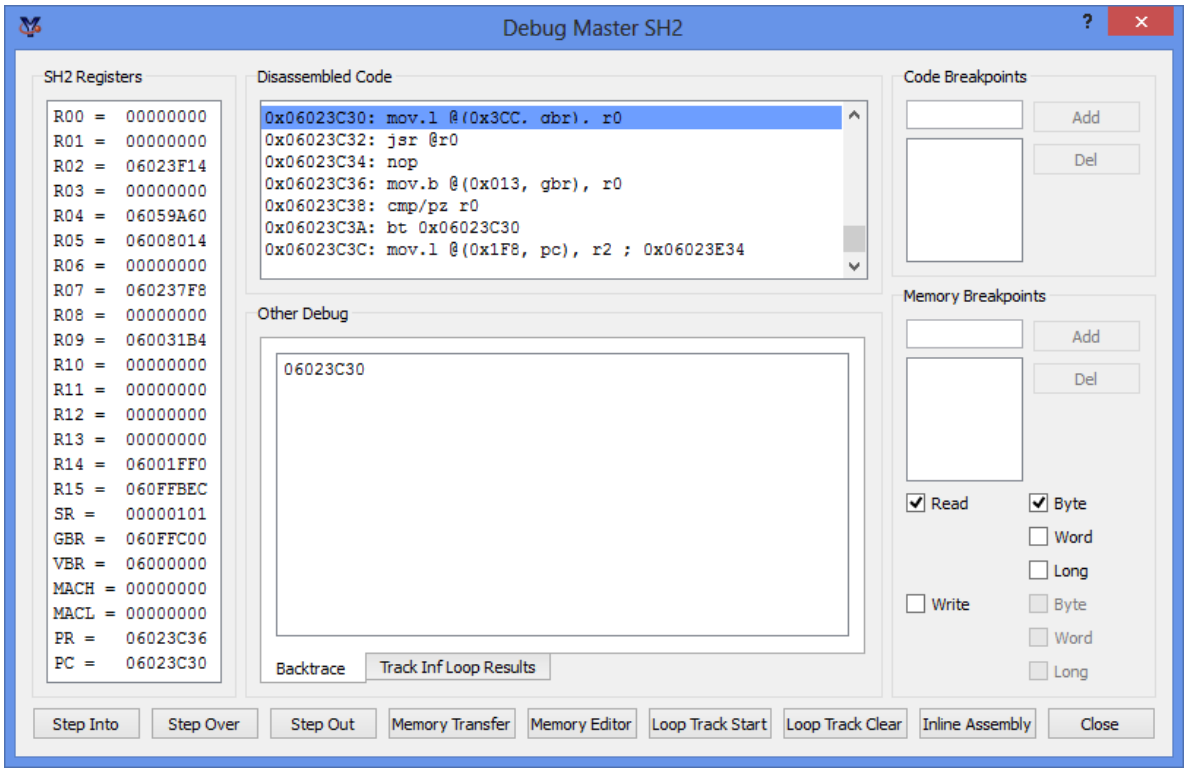
きっとあなたはこの質問をすることになるでしょう、幸いなことに、Yabauseエミュレータの設計者は、プロのデバッグを実行できるようにするために、あるいは公式のSDKに近いことができるツールのセットに追加しました。

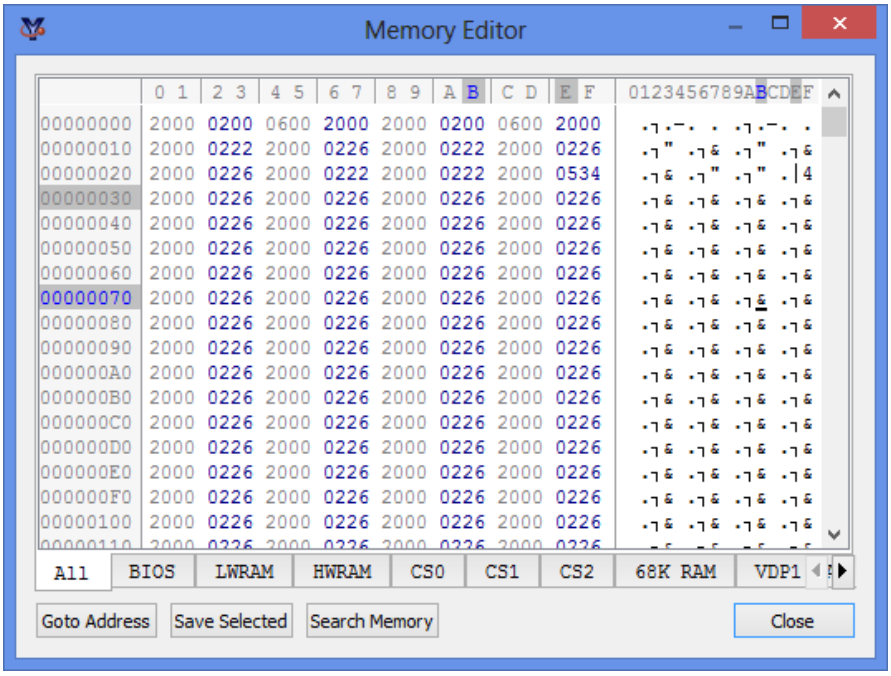
All those wonderful tools to debug, you can find them in the "Debug" menu of the Yabause:

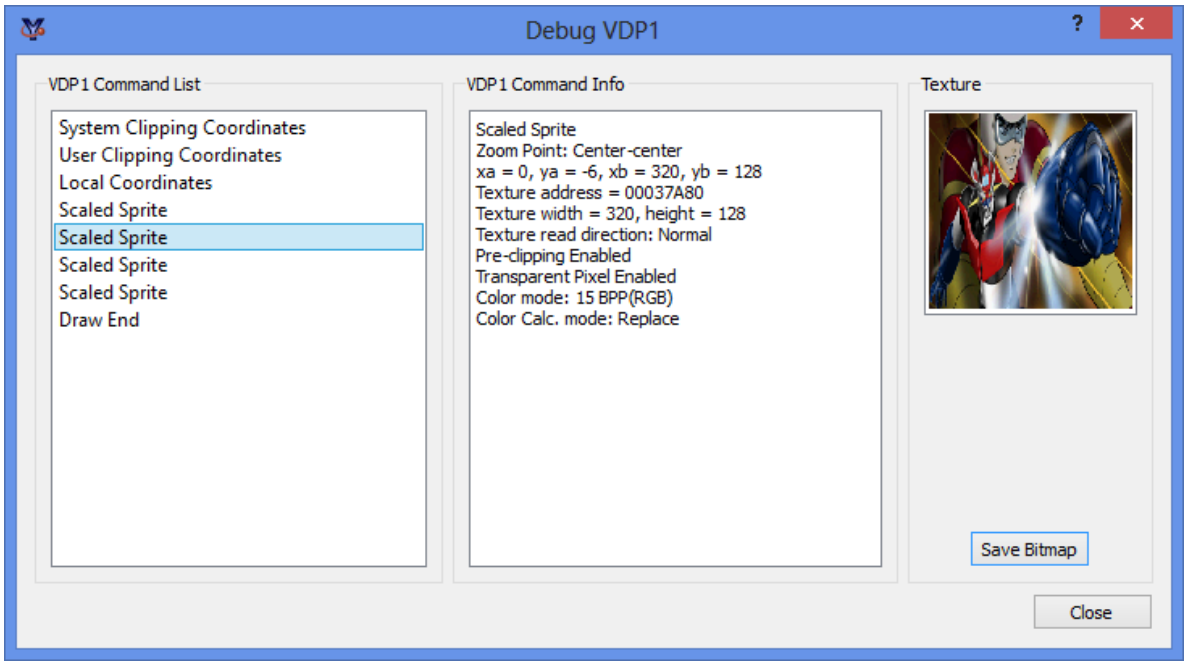
The menu contains a series of viewers, with which we can observe from the contents of the registers of a SH-2 CPU, up to data referring to the video chips.

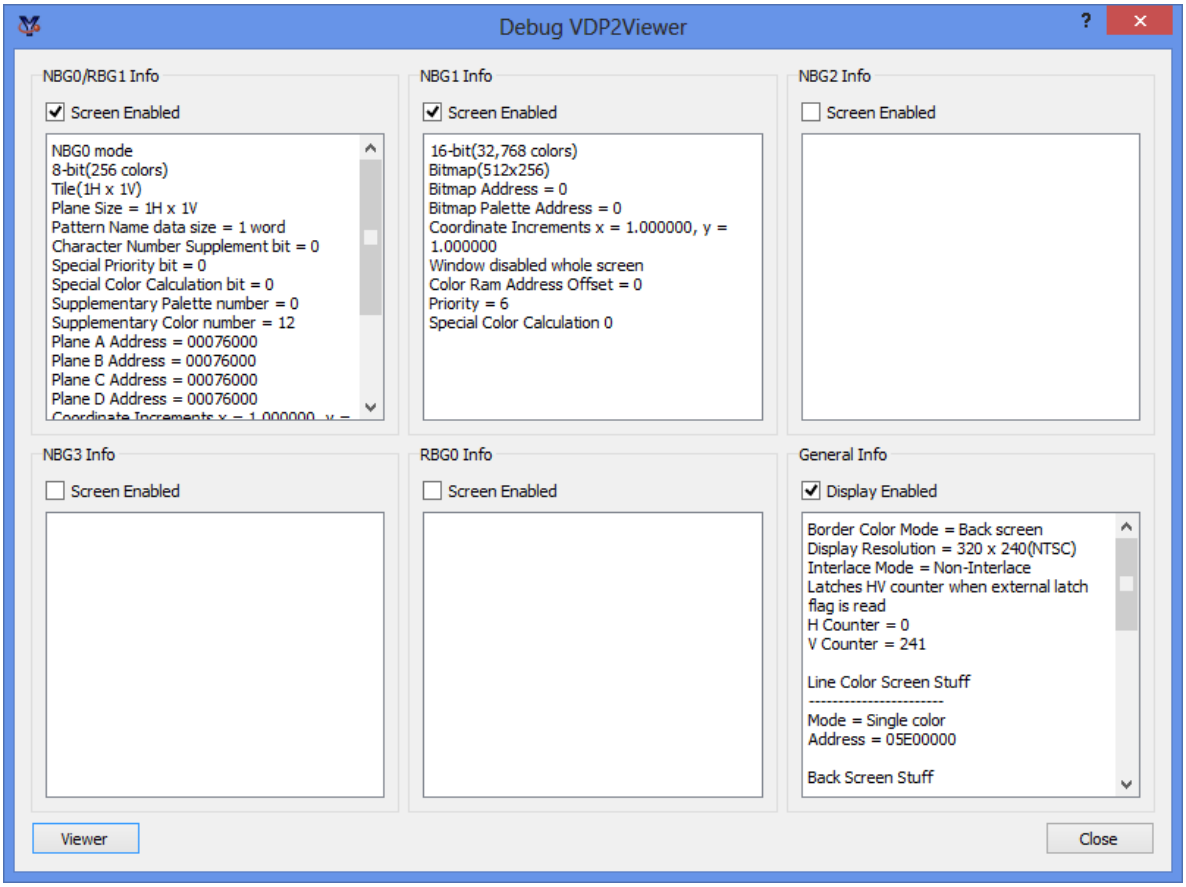
このメニューには、SH-2 CPUのレジスタの内容からビデオチップを参照するデータまで、一連のビューアがあります。

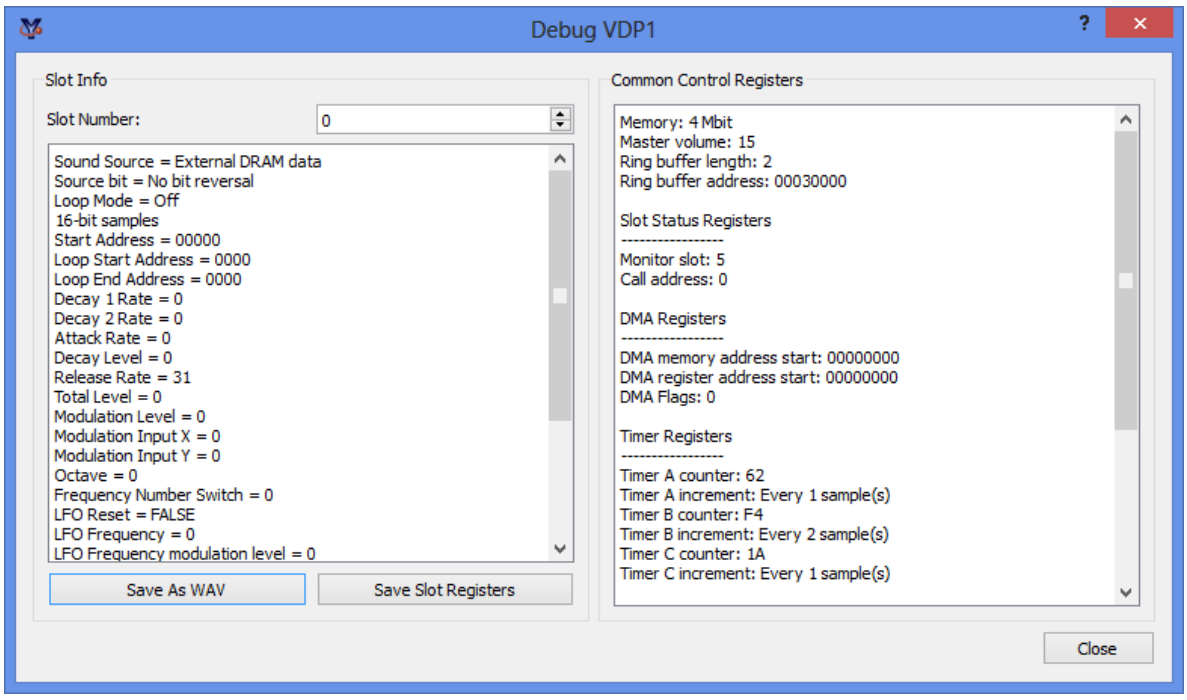

Debug M68K
M68K Registers
D0 = 00000000
D1 = 4E710000
D2 = 000000E0
D3 = 00000000
D4 = 00000000
D5 = 00000000
D6 = 0000FFFF
D7 = 00000010
A0 = 00000780
A1 = 00009C00
A2 = 00009000
A3 = 000088A0
A4 = 000083C0
A5 = 001001E0
A6 = 00007000
A7 = 00009FF4
SR = 00002004
PC = 00001570
Disassembled Code
01570: bmi.w 15C2
01574: moveq #0x0, d5
01576: move.b 0x2(a4), d5
0157A: lsl.w
0157C: ori to CCR
0157E: ori
01580: move.w 0x8(a6, Xn), d5
01584: move.l 0x8(a4), d0
01588: bne.w 1590
0158C: bra.w 15C2
01590: movea.l d0, a2
01592: btst #0x5, 0x0(a4)
01598: beq.w 15A0
0159C: bsr.w 15D4
015A0: btst #0x6, 0x0(a4)
015A6: beq.w 15AE
015AA: bsr.w 167A
015AE: cmp.w 0x38(a4), d5
015B2: bne.w 15BA
015B6: bra.w 15C2
015BA: move.w d5, 0x38(a4)
015BE: bsr.w 3AF6
015C2: adda.w #0x40, a4
015C6: adda.w #0x20, a5
015CA: dbf d7, 156C
015CE: movem
Code Breakpoints
Add
Del
Step Into
Memory Transfer
Memory Editor
Close


Debug Master SH2
SH2 Registers
R00 = 00000000
R01 = 00000000
R02 = 06023F14
R03 = 00000000
R04 = 06059A60
R05 = 06008014
R06 = 00000000
R07 = 060237F8
R08 = 00000000
R09 = 060031B4
R10 = 00000000
R11 = 00000000
R12 = 00000000
R13 = 00000000
R14 = 06001FF0
R15 = 060FFBEC
SR = 00000101
GBR = 060FFC00
VBR = 06000000
MACH = 00000000
MACL = 00000000
PR = 06023C36
PC = 06023C30
Disassembled Code
0x06023C30: mov.l @(0x3CC, gbr), r0
0x06023C32: jsr @r0
0x06023C34: nop
0x06023C36: mov.b @(0x013, gbr), r0
0x06023C38: cmp/pz r0
0x06023C3A: bt 0x06023C30
0x06023C3C: mov.l @(0x1F8, pc), r2 ; 0x06023E34
Code Breakpoints
Add
Del
Other Debug
06023C30
Backtrace
Track Inf Loop Results
Memory Breakpoints
Add
Del
Read
Byte
Word
Long
Write
Byte
Word
Long
Step Into
Step Over
Step Out
Memory Transfer
Memory Editor
Loop Track Start
Loop Track Clear
Inline Assembly
Close


Memory Editor
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
00000000 2000 0200 0600 2000 2000 0200 0600 2000
00000010 2000 0222 2000 0226 2000 0222 2000 0226
00000020 2000 0226 2000 0222 2000 0222 2000 0534
00000030 2000 0226 2000 0226 2000 0226 2000 0226
00000040 2000 0226 2000 0226 2000 0226 2000 0226
00000050 2000 0226 2000 0226 2000 0226 2000 0226
00000060 2000 0226 2000 0226 2000 0226 2000 0226
00000070 2000 0226 2000 0226 2000 0226 2000 0226
00000080 2000 0226 2000 0226 2000 0226 2000 0226
00000090 2000 0226 2000 0226 2000 0226 2000 0226
000000A0 2000 0226 2000 0226 2000 0226 2000 0226
000000B0 2000 0226 2000 0226 2000 0226 2000 0226
000000C0 2000 0226 2000 0226 2000 0226 2000 0226
000000D0 2000 0226 2000 0226 2000 0226 2000 0226
000000E0 2000 0226 2000 0226 2000 0226 2000 0226
000000F0 2000 0226 2000 0226 2000 0226 2000 0226
00000100 2000 0226 2000 0226 2000 0226 2000 0226
All
BIOS
LWRAM
HWRAM
CS0
CS1
CS2
68K RAM
VDP1
Goto Address
Save Selected
Search Memory
Close


Debug VDP1
VDP1 Command List
System Clipping Coordinates
User Clipping Coordinates
Local Coordinates
Scaled Sprite
Scaled Sprite
Scaled Sprite
Scaled Sprite
Draw End
VDP1 Command Info
Scaled Sprite
Zoom Point: Center-center
xa = 0, ya = -6, xb = 320, yb = 128
Texture address = 00037A80
Texture width = 320, height = 128
Texture read direction: Normal
Pre-clipping Enabled
Transparent Pixel Enabled
Color mode: 15 BPP(RGB)
Color Calc. mode: Replace
Texture
Save Bitmap
Close


Debug VDP2Viewer
NBG0/RBG1 Info
Screen Enabled
NBG0 mode
8-bit(256 colors)
Tile(1H x 1V)
Plane Size = 1H x 1V
Pattern Name data size = 1 word
Character Number Supplement bit = 0
Special Priority bit = 0
Special Color Calculation bit = 0
Supplementary Palette number = 0
Supplementary Color number = 12
Plane A Address = 00076000
Plane B Address = 00076000
Plane C Address = 00076000
Plane D Address = 00076000
NBG1 Info
Screen Enabled
16-bit(32,768 colors)
Bitmap(512x256)
Bitmap Address = 0
Bitmap Palette Address = 0
Coordinate Increments x = 1.000000, y = 1.000000
Window disabled whole screen
Color Ram Address Offset = 0
Priority = 6
Special Color Calculation 0
NBG2 Info
Screen Enabled
NBG3 Info
Screen Enabled
RBG0 Info
Screen Enabled
General Info
Display Enabled
Border Color Mode = Back screen
Display Resolution = 320 x 240(NTSC)
Interlace Mode = Non-Interlace
Latches HV counter when external latch flag is read
H Counter = 0
V Counter = 241
Line Color Screen Stuff
Mode = Single color
Address = 05E00000
Back Screen Stuff
Viewer
Close


Debug VDP1
Slot Info
Slot Number:
0
Sound Source = External DRAM data
Source bit = No bit reversal
Loop Mode = Off
16-bit samples
Start Address = 00000
Loop Start Address = 0000
Loop End Address = 0000
Decay 1 Rate = 0
Decay 2 Rate = 0
Attack Rate = 0
Decay Level = 0
Release Rate = 31
Total Level = 0
Modulation Level = 0
Modulation Input X = 0
Modulation Input Y = 0
Octave = 0
Frequency Number Switch = 0
LFO Reset = FALSE
LFO Frequency = 0
LFO Frequency modulation level = 0
Save As WAV
Save Slot Registers
Common Control Registers
Memory: 4 Mbit
Master volume: 15
Ring buffer length: 2
Ring buffer address: 00030000
Slot Status Registers
Monitor slot: 5
Call address: 0
DMA Registers
DMA memory address start: 00000000
DMA register address start: 00000000
DMA Flags: 0
Timer Registers
Timer A counter: 62
Timer A increment: Every 1 sample(s)
Timer B counter: F4
Timer B increment: Every 2 sample(s)
Timer C counter: 1A
Timer C increment: Every 1 sample(s)
Close

As you can see in the previous windows, you could use the function Debug (void) and the function Exec_Assembler_Hitachi_SH2_CPU (void). From the source code of Mazinger Z Homebrew, to modify the records of the SH-2 CPU and debug them using these magnificent Yabause tools.

```
58  // この関数は日立SH-2 CPUのテストを実行することを可能にし、SDKがSEGA Saturnハードウ
59  // ェアへのフルアクセスでCOFFフォーマットでバイナリをコンパイルすることを可能にすることを実証します。
60
61
62      asm ("      .ORG    H'1006          ");
63      asm ("      MOV     #H'7B1 , R1     ");  // STAGES=3 CYCLES=1
64      asm ("      MOV     #H'7C4 , R5     ");  // STAGES=3 CYCLES=1
65      asm ("      NOT        R5  , R1     ");
66      asm ("      ADD     #-1    , R0     ");  // STAGES=3 CYCLES=1
67      asm ("      TST     R0     , R0     ");  // STAGES=6 CYCLES=1
68      asm ("      SETT                    ");
69      asm ("      NEG     R1     , R0     ");
70      asm ("      .ALIGN  2               ");
71      asm ("      CLRMAC                  ");
72      asm ("      CLRT                    ");
73      asm ("DATAS1:                       ");
74      asm ("      MOV     #H'4D3 , R0     "); // STAGES=3 CYCLES=1
75      asm ("      MOV     #H'7C3 , R1     "); // STAGES=3 CYCLES=1
76      asm ("OPERATION:                    ");
77      asm ("      ADD        R0 , R1      "); // STAGES=3 CYCLES=1
78      asm ("      NOT        R0 , R1      ");
79      asm ("      STS       MACH , R1     "); // STAGES=4 CYCLES=1
80      asm ("      SUB        R0 , R1      ");
81      asm ("      MULU       R2 , R0      "); // STAGES=6/7*3 CYCLES=1
82      asm ("NOP                           ");
83      asm ("TARGET:                       ");
84      asm ("      .ALIGN  4               ");
85      asm ("      SETT                    ");
86      asm ("      OR         R5 , R0      ");
87      asm ("DATAS2:                       ");
88      asm ("      MOV     #H'00A , R0     "); // STAGES=3 CYCLES=1
89      asm ("      MOV     #H'01C , R1     "); // STAGES=3 CYCLES=1
90      asm ("      ADD        R5 , R1      "); // STAGES=3 CYCLES=1
91      asm ("      LDC        R1 , SR      ");
92      asm ("      STS       MACL , R0     "); // STAGES=4 CYCLES=1
93      asm ("      SETT                    ");
```

前のウィンドウでわかるように、Debug（void）関数とExec_Assembler_Hitachi_SH2_CPU（void）関数を使用できます。 Mazinger Z Homebrewのソースコードから、SH-2 CPUのレコードを修正し、これらのすばらしいYabauseツールを使用してそれらをデバッグします。

MAZINGER Z HOMEBREW

マジンガーz

セガサターンで動作するようにプログラムされたアクションゲーム。

文書の終わり

NOVA32 DEVELOPMENT
SOFTWARE