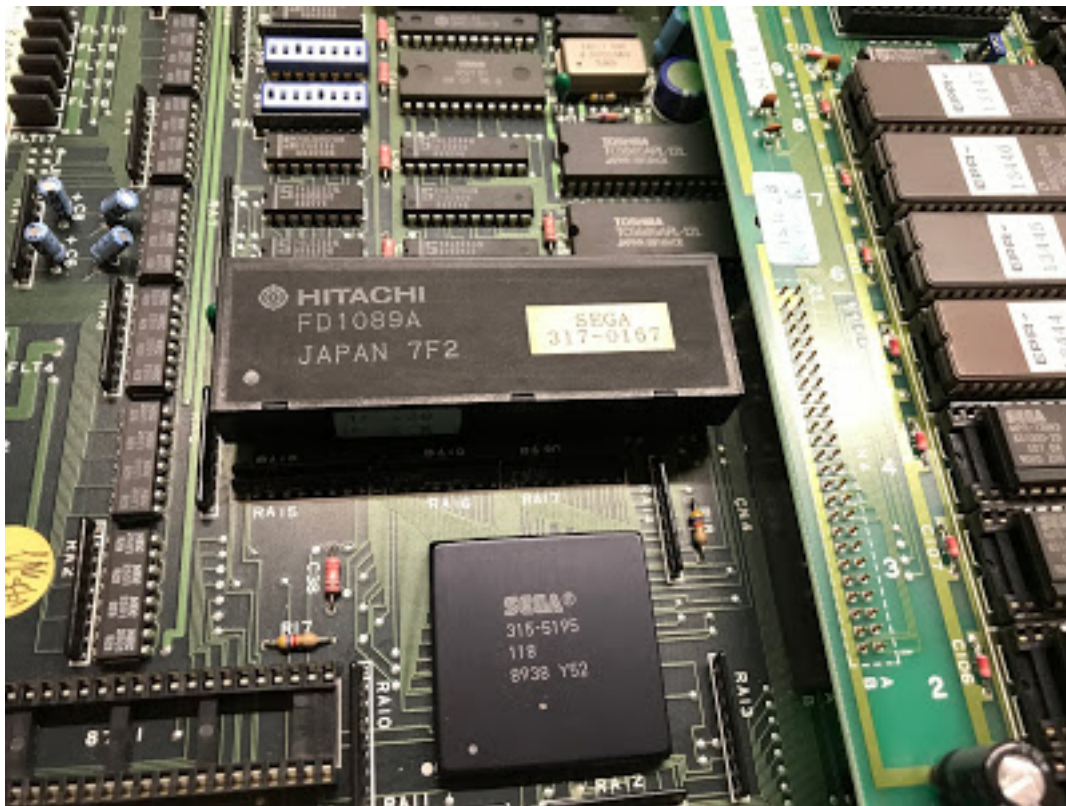


A live online copy of this guide can be found there:

<https://arcadehacker.blogspot.com/2018/12/sega-system16-security-programming-guide.html>

Dear all, after some lengthy testing we are happy to release full details on the security programming of the Hitachi FD1089 / FD1094 cpus used in Sega's System 16 / 18 / 24 / X motherboards.



This guide is the result of several years of work by a small group of arcade enthusiasts to unravel the secrets of the security implementation found in one of the most loved and popular arcade game platforms. Thanks to this work it is now possible to fully preserve most Sega 16 bit systems enabled with security as fully working unmodified originals.

Unlike previous projects this time we recommend the usage of a dedicated pcb to interface with the chips due to the high pin count involved in the programming.

Additional details of the inner workings of Sega's FD security modules will be published over time in this blog. Work on the earlier MC 8 bit modules used in some System 1 / 2 boards and sound of Sega 16 is still in progress and will be published when completed.

Thanks to everyone who has helped make this a reality including all kind donors and testers.

Sega Hitachi FD1089 / FD1094 Security Programming Guide

This document will guide you through the basics of preparing your setup and testing the a clean desuicide method on any known Sega 16 / 18 / 24 / X board revisions using Hitachi FD1089 or FD1094 modules. You can find a pdf copy of this guide and code on the following link:

https://github.com/ArcadeHacker/ArcadeHacker_Sega_Hitachi

What's needed

Arduino programmer hardware

- 1x Arduino Mega 2560 rev 3 (with USB cable)
 - <https://store.arduino.cc/arduino-mega-2560-rev3>
- 3x 40 Pin Male Single Row Pin Header 2.54mm spacing
 - <https://www.ebay.com/itm/10PCS-40Pin-2-54mm-Male-PCB-Single-Row-Straight-Header-Strip-Connector-Arduino/371644788140>
- 1x 64 pin IC socket 2.54mm spacing, a couple of options:
 - Turned pin socket:
 - <https://www.ebay.com/itm/1-X-GOLD-64P-Socket-DIP-PIN-64-22-86mm-gold-plated-polyester-UL94V-0-1A/113372793987>
 - ZIF socket:
 - <https://www.ebay.com/itm/1pc-64Pin-64-Pin-2-54mm-IC-Test-Universal-ZIF-Socket/163152882218>
- 1x ArcadeHacker FD1089 / FD1094 arduino mega shield pcb

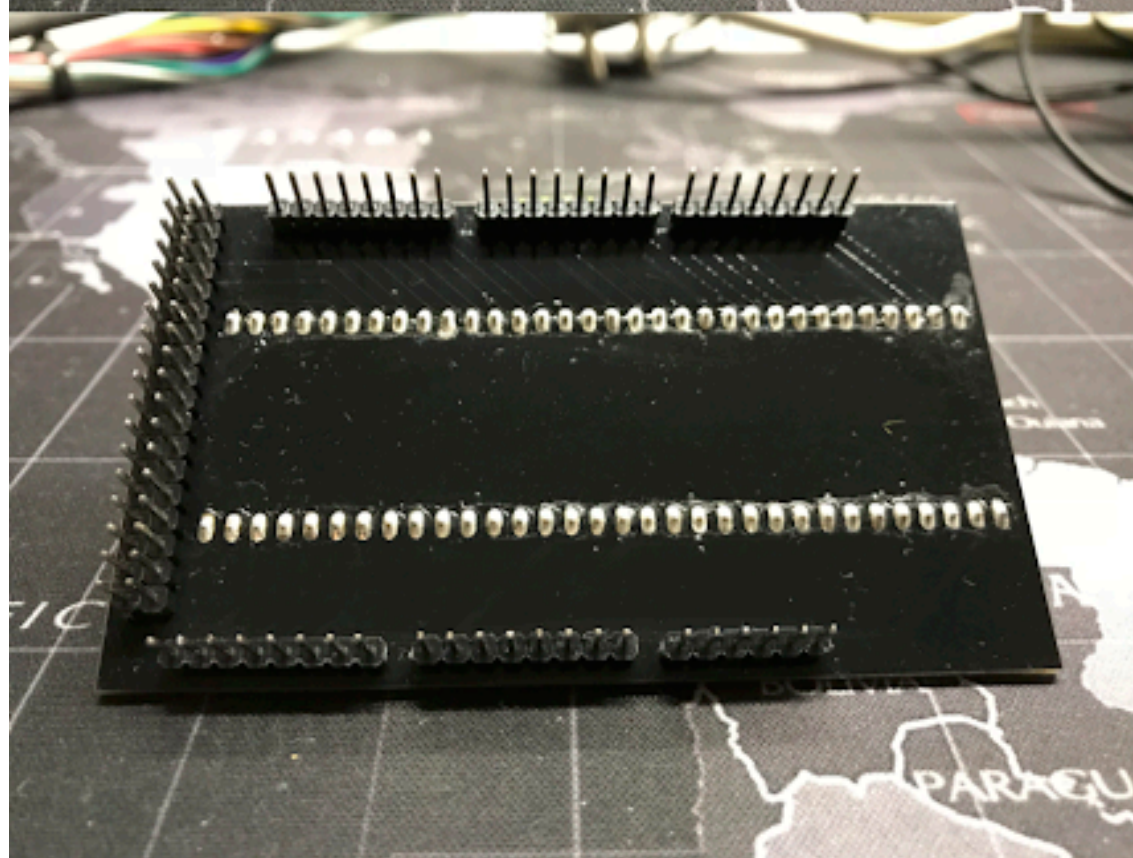
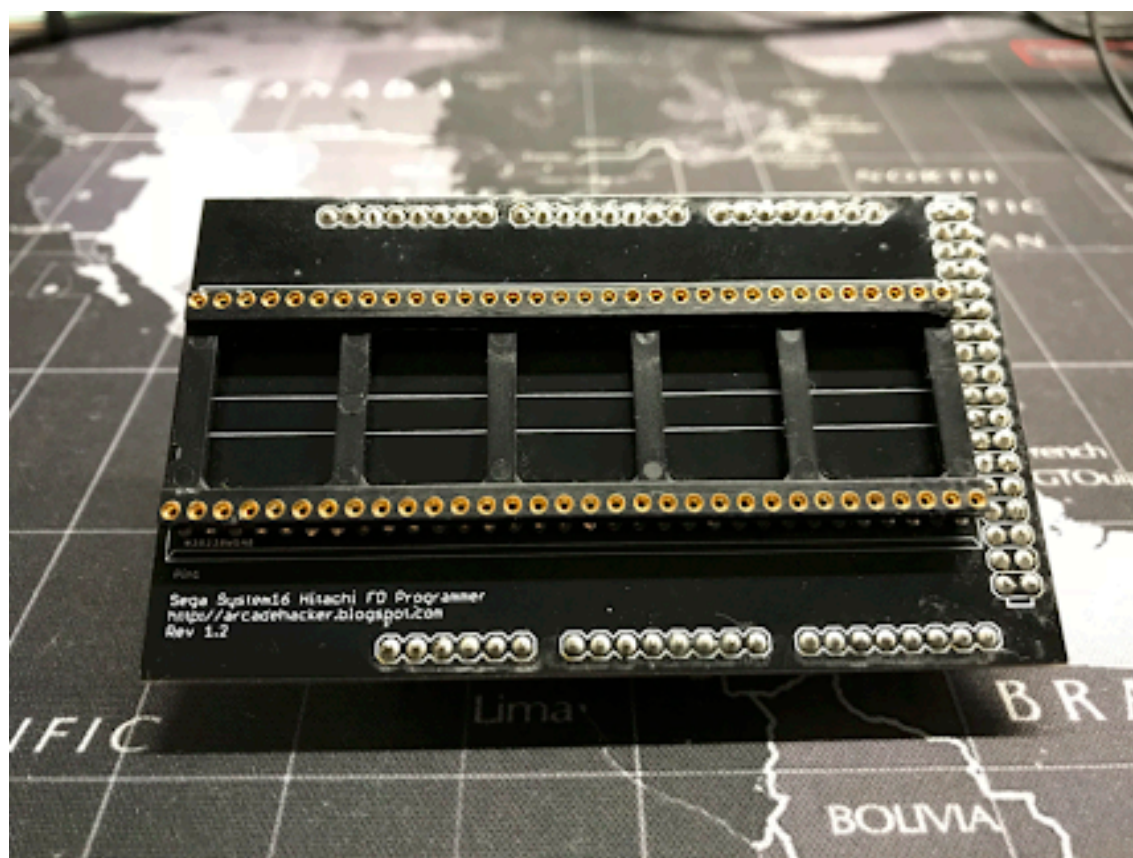
- Build / order your own:
 - Pcbway.com 72h on demand fabrication for \$5
 - https://www.pcbway.com/project/shareproject/W38238MSJ9_W38238MSN8_SegaFD_2560v2.html
- Design / customize your own:
 - PCB design source files
 - Eagle
 - https://github.com/ArcadeHacker/ArcadeHacker_Sega_Hitachi/blob/master/Eagle_SegaFD_2560V2.zip
 - Gerbers
 - https://github.com/ArcadeHacker/ArcadeHacker_Sega_Hitachi/blob/master/Gerbers_SegaFD_2560v2.zip
- Soldering iron and solder

Software

- Computer with Arduino IDE Software
 - <https://www.arduino.cc/en/Main/Software>
- ArcadeHacker Sega Hitachi FD1089 or FD1094 Arduino programs
 - https://github.com/ArcadeHacker/ArcadeHacker_Sega_Hitachi

Assembling and preparing your Arduino programmer

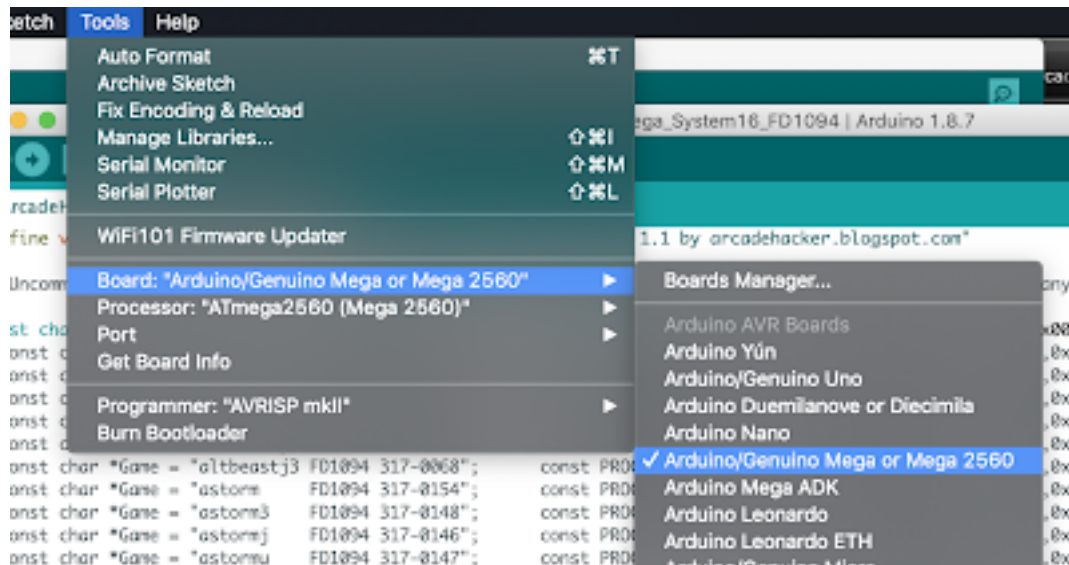
This step is pretty straight forward, solder all the pin headers to your programmer pcb as well as the IC 64pin socket. The fully assembled pcb looks like this:



After the pcb shield is ready just sandwich it together with your Arduino Mega 2650.



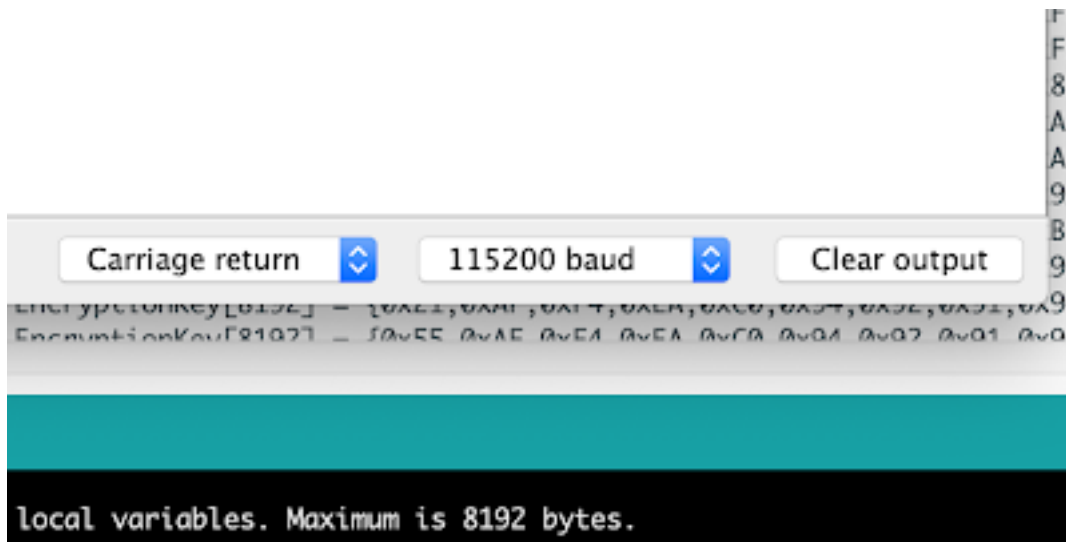
Download and install software for your OS
from <https://www.arduino.cc/en/Main/Software>
Select the right arduino board type before connecting your arduino to your computer.
Tools -> Board -> Mega 2560.



Connect the Arduino to your computer by plugin the USB cable and choose the correct serial port in Tools -> Port.

Open the ArcadeHacker Sega FD1094 or FD1089 .ino file in the Arduino environment. Compile and Upload the sketch to the Arduino unit.

Open the Arduino IDE serial monitor found in Tools -> Serial Monitor. Configure the two settings found in the lower right part of the serial monitor window as follows: Carriage Return and 115200 bauds.



Close and open the serial monitor, you should now see the following text on screen. If this is the case you have successfully setup and configured your Arduino programmer.



```
Arduino
/dev/cu.usbmodem141401 (Arduino/Genuino Uno)

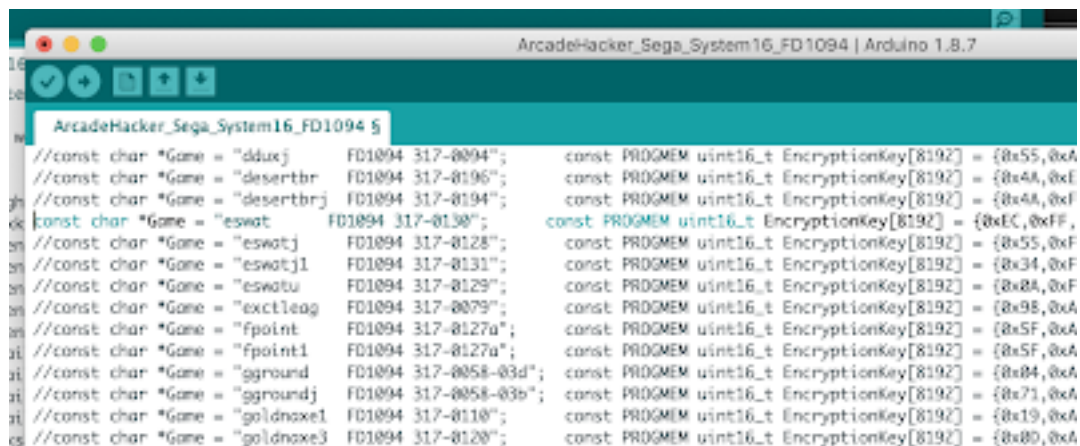
Sega System16/18/24/X FD1094 Security Programmer 1.1 by arcadehacker.blogspot.com

Commands:
w - Writes encryption key data into the FD1094 chip. Select your game by uncommenting the appropriate line in the arduino program source code.
v - Dumps and compares a parity bit for all encryption data bytes stored inside the FD1094 chip.

Game configuration selected: BLANK XXXXXX XXX-XXXX
```

Programming security keys

Look inside the Arduino FD1089 or FD1094 code for the game you intend to program the encryption keys into the FD chip and uncomment the desired line of code. Only one game can be uncommented at any time, don't forget to comment the default blank game setting at the beginning of the list.



```
ArcadeHacker_Sega_System16_FD1094
//const char *Game = "dduxj" FD1094 317-8094"; const PROGMEM uint16_t EncryptionKey[8192] = {0x55,0xA
//const char *Game = "desertbr FD1094 317-8196"; const PROGMEM uint16_t EncryptionKey[8192] = {0x4A,0xE
//const char *Game = "desertbrj FD1094 317-8194"; const PROGMEM uint16_t EncryptionKey[8192] = {0x4A,0xF
const char *Game = "eswat" FD1094 317-8130"; const PROGMEM uint16_t EncryptionKey[8192] = {0xEC,0xFF,
//const char *Game = "eswatj FD1094 317-8128"; const PROGMEM uint16_t EncryptionKey[8192] = {0x55,0xF
//const char *Game = "eswatj1 FD1094 317-8131"; const PROGMEM uint16_t EncryptionKey[8192] = {0x34,0xF
//const char *Game = "eswatv FD1094 317-8129"; const PROGMEM uint16_t EncryptionKey[8192] = {0x8A,0xF
//const char *Game = "exctleag FD1094 317-8079"; const PROGMEM uint16_t EncryptionKey[8192] = {0x9B,0xA
//const char *Game = "fpoint FD1094 317-8127a"; const PROGMEM uint16_t EncryptionKey[8192] = {0x5F,0xA
//const char *Game = "fpoint1 FD1094 317-8127a"; const PROGMEM uint16_t EncryptionKey[8192] = {0x5F,0xA
//const char *Game = "gground FD1094 317-8058-83d"; const PROGMEM uint16_t EncryptionKey[8192] = {0x04,0xA
//const char *Game = "ggroundj FD1094 317-8058-83b"; const PROGMEM uint16_t EncryptionKey[8192] = {0x71,0xA
//const char *Game = "goldnoxel FD1094 317-8110"; const PROGMEM uint16_t EncryptionKey[8192] = {0x19,0xA
//const char *Game = "goldnoxel3 FD1094 317-8120"; const PROGMEM uint16_t EncryptionKey[8192] = {0x8D,0xA
```

If the above was done correctly you should now compile and upload the program to your Arduino without problems. Once the upload process is finished open the serial monitor found in Tools. The program prompt should display the game you have configured.

```
/dev/cu.usbmodem141401 (Ar

Sega System16/18/24/X FD1094 Security Programmer 1.1 by arcadehacker.blogspot.com

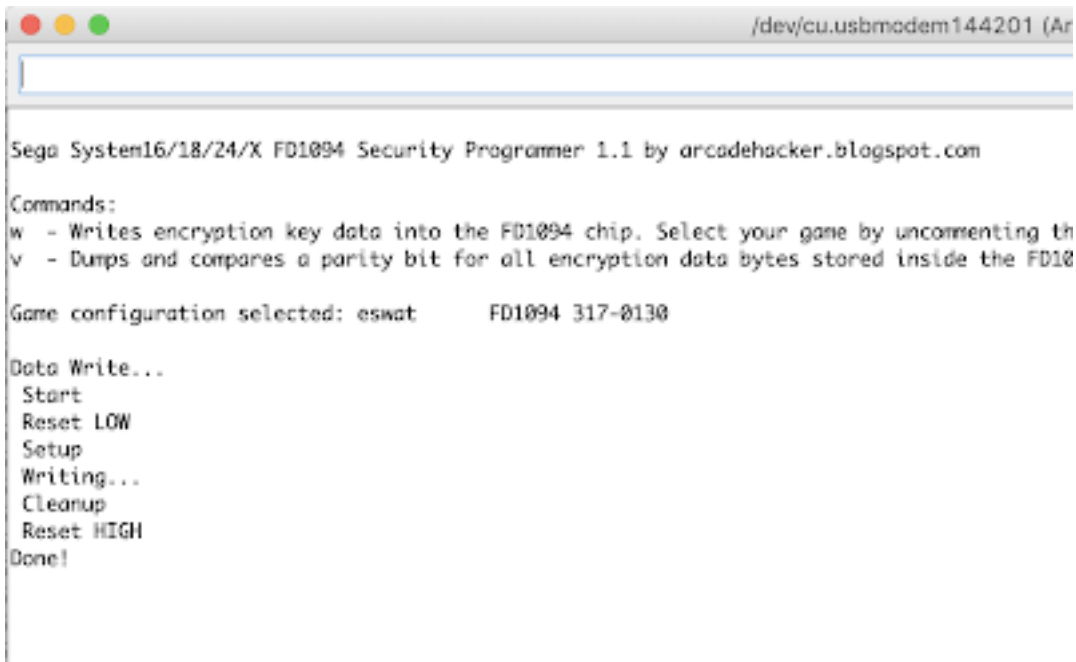
Commands:
w - Writes encryption key data into the FD1094 chip. Select your game by uncommenting th
v - Dumps and compares a parity bit for all encryption data bytes stored inside the FD10

Game configuration selected: eswat      FD1094 317-0130 ←
```

For this example we have selected "eswat" a System 16 game using the Hitachi FD1094 security module "317-0160". FD1094 security modules can be repurposed for any game and it is not required that a label in the cpu module matches the intended game security key to be programmed. This will not the case with FD1089 modules as two different encryption schemes exist (FD1089A & FD1089B): A variants of the chip are only compatible with games roms for other A variants, and B variants for B games only.



Having inserted the module in the socket we can proceed to program the security keys by typing w and pressing enter. Make sure you replace the module battery if necessary before attempting a key load, a new battery will last at least 20-30 years so don't expect to have to repeat this often. Once the process finishes you can disconnect the Arduino module from your computer and remove the cpu module.

A screenshot of a terminal window with a title bar showing three colored buttons (red, yellow, green) and the path /dev/cu.usbmodem144201 (Ar... The terminal displays the following text:

```
Sega System16/18/24/X FD1094 Security Programmer 1.1 by arcadehacker.blogspot.com

Commands:
w - Writes encryption key data into the FD1094 chip. Select your game by uncommenting th
v - Dumps and compares a parity bit for all encryption data bytes stored inside the FD10

Game configuration selected: eswat      FD1094 317-0130

Data Write...
Start
Reset LOW
Setup
Writing...
Cleanup
Reset HIGH
Done!
```

If you would like to verify that the contents of the cpu match the intended configuration loaded in your programmer you can perform a verification by typing v and pressing enter. This verification compares a 1 bit parity per byte calculated internally by the security module against the encryption configuration in your Arduino.

WARNING: This verification procedure is 100% safe in FD1094 modules. Unfortunately this is not the case with FD1089 modules as the verification will delete the data inside your module, please do not attempt verification with any valuable FD1089 modules especially if currently undumped or not preserved in Mame.

Final words

This is all there is to preserving your Sega FD1089 / FD1094 modules as working units, we hope this milestone will help you and the rest of the arcade community preserve your loved games as originals and stop the general discarding of Sega Hitachi modules.

As mentioned in the opening of this post, a number of follow up articles in this blog will reveal the inner workings and curiosities of these artfully crafted security modules.

Happy preservation.