Hello, Guest.

With the recent attacks against members, and lack of care towards their mental health we have started a fundraiser to help fund the life saving charity the Samaritans. Without their support and efforts many people we know wouldn't be alive.

If you're able to help, you can donate via our fundraiser on GoFundMe Link

Thank you.

PS1 MFO mod on SCPH-750x PS1s (my e-Journal edition)

provato · Later today at 8:29 PM

Later today at 8:29 PM Thread starter #1

P

provato
Registered

Registered

Joined: Nov 8, 2019
Messages: 23
Reaction score: 7

*Note to the reader:
The following text is taken directly from my personal electronic journal about various electronics (e² Journal), so it has many references and reminders to myself.
Please bare with this peculiar writing if you are interested in the subject I'm writing about, as it contains many valuable details.

13/08/20

Dear e² Journal,

_This is a big step for me. It is the third time I investigate the **DFO** (**D**ual **F**requency **O**scillator) mod for various video game systems, but this time I am confident I can manage building the programmer and the mod (at least for my ps1). As always, before I install a mod/ upgrade on any of my consoles, it's important to remember the basic principles I have set:

- Aim for perfect RGB video output, but maintain RF, Composite, S-video and Component video capabilities (if these were available from factory)
- Aim for perfect NTSC 60Hz/29,97fps and perfect PAL 50Hz/25fps signal out of a PAL console
- Create completely and easily reversible mods

- NO TRACE CUTS! (or at least very easily reversible trace cuts when they are necessary)
- NO CONSOLE CUTS! (very minor holes that respect the console's appearance can be acceptable though)
- Maintain original media compatibility (CDs, DVDs, Cartridges, Cards etc)
- Aim for home reproduction media playability (CD-Rs, DVD-Rs, Repro carts etc)
- Aim for external storage media playability (Roms/Disc Image Files stored in SD cards, SSDs, HDDs etc)

_First of all, the websites/forums that contain the information needed for building these mods are three:

1. https://nfggames.com/forum2/index.php?topic=5744.0
2. https://circuit-board.de/forum/index.php/Thread/18016-DFO-Dual-Frequency-Oscillator/, and
3. https://immerhax.com/?p=111

I intend to build the **MFO** (**M**ultiple **F**requency **O**scillator) first for my SCPH-7502 playstation, as this particular mod provides all 3 correct frequencies/clocks (CPU, GPU & Colour Subcarrier) for both NTSC and PAL signals after autodetecting them.
_Second is the parts list. The plan is to buy parts so that I can build and program my own **CDCE9XX programmer** (3 of them) and also build and program at least 5 MFOs (I'll keep one or two and sell the rest to hopefully amortize the parts' cost).

→ Table 1: CDCE9XX Programmer Parts List

Quantity	Name/Mfr No.	Store	Price (+ Shipping & VAT)
3	CDCE9XX Programmer PCB	oshpark.com/profiles/micro	12,42 €
3	FT232RL-REEL USB to Serial UART Interface IC	mouser.com	12,12 €
3	ATMEGA48A-AU Microcontroller	mouser.com	4,16 €
3	EDAC 690-005-299-043 5-pin Mini USB-B Right Angle SMT Connector	mouser.com	2,20 €
3	Alps Alpine SSSS810701 Horizontal 1 pole-2 position-1.1	mouser.com	3,80 €

	knob-1.5 travel Slide Switch		
3	CTS ATS080B 8MHz 18pF THT Crystal	mouser.com	1,60 €
3	OSRAM LS R976-NR-1 SMD LED 633nm Red 20mA 0805	mouser.com	1,39 €
9	KEMET C0805C104K5RACTU SMD 100nF 10% 0805 MLC Capacitor	mouser.com	1,90 €
3	AVX 08053D475KAT2A SMD 4.7µF 10% 0805 MLC Capacitor	mouser.com	2,08 €
6	Vishay VJ0805A180GXJCW1BC SMD 18pF 2% 0805 MLC Capacitor	mouser.com	1,65 €
3	Vishay CRCW0805330RFKEA SMD 330Ω 1% 0805 Resistor	mouser.com	0,99 €
6	Vishay CRCW080510K0JNEA SMD 10kΩ 5% 0805 Resistor	mouser.com	1,19 €
6	Vishay CRCW08054K70FKEA SMD 4.7kΩ 1% 0805 Resistor	mouser.com	1,19 €
3	*KOA Speer SPR1CT52R220J THT 22Ω 5% Axial Resistor	mouser.com	1,44 €
20	2x3 Straight Pin 2.54 pitch IDC Male Box Socket/Header/ Connector (ebay item No. 301819338155)	ebay.com seller: gc_supermarket	2,87 €
20	2x3 Pin 2.54 pitch IDC Female Socket/Header/ Connector	ebay.com seller: gc_supermarket	2,87 €

	(ebay item No. 302946224212)		
1	ATtiny44 USBTinyISP AVR Programmer (ebay item No. 302189016479)	ebay.com seller: gc_supermarket	3,06 €
40	4 Straight Pin JST Male & Female Connectors (Set) (ebay item No. 292874607958)	ebay.com seller: gc_supermarket	4,25 €
1	Ribbon IDC Multicolor Cable Roll 1m 40way	ebay.com seller: gc_supermarket	2,75 €
		TOTAL	**63,93 €**

* This 22Ω THT resistor must be connected in series with the voltage supply (VTG/V_TARGET) to the DFO/MFO. I'll solder this resistor on the ribbon cable and avoid butchering the Programmer PCB.

_Cost per Programmer Boards (CDCE9XX & USBTinyISP):
(12,42/3) + (12,12/3) + (4,16/3) + (2,2/3) + (3,8/3) + (1,6/3) + (1,39/3) + (1,9/3) + (2,08/3) + (1,65/3) + (0,99/3) + [(1,19x2)/3] + (1,44/3) + [(2,87x2)x(2/20)] + 3,06 + [4,25x(4/40)] + (2,75/3) = **21,02 €**

→ Table 2: MFO Parts List

Quantity	Name/Mfr No.	Store
6	MFO PCB (Rev. B)	oshpark.com/shared_projects/6oHbyT4x
5	T.I. CDCE925PWR 2-PLL Programmable Clock Synthesizer	mouser.com
5	T.I. TPS78218DDCT 1.8V LDO Voltage Regulator	mouser.com
5	ABRACON ABLS3-27.000MHZ-D4YF-T 2.7MHz 18pF SMD Low Profile Crystal	mouser.com

10	Vishay CRCW060318R0FKEA SMD 18Ω 1% 0603 Resistor	mouser.com
10	Vishay CRCW060310K0FKEA SMD 10kΩ 1% 0603 Resistor	mouser.com
20	Yageo CC0603KRX5R8BB105 SMD 1µF 10% 0603 MLC Capacitor	mouser.com
		TOTAL

_**Cost per MFO Board:** (3,97/6) + (19,78/5) + (4,62/5) + (3.05/5) + (0,99/5) + (0,97/5) + (1,70/5) ≃ **6,89 €**

_Now, I'll just have to wait for all these parts to arrive and start assembling and programming. In the meantime I downloaded all the necessary software for the CDCE9XX Programmer & the MFO boards:

1. *(Windows drivers for USBTinyISP)* adafruit_drivers_2.4.0.0.exe
   LINK: https://learn.adafruit.com/usbtinyisp/drivers
2. *(AVRDUDESS GUI)* AVRDUDESS-2.13-setup.exe
   LINK: https://blog.zakkemble.net/avrdudess-a-gui-for-avrdude
3. *(CDCE9XX Programmer Firmware)* CDCE9XXX_PROGRAMMER_FW_v1.zip
   LINK: https://circuit-board.de/forum/inde...p/?s=931901dba62166cf9de532e05e8179967696b698
4. *(CDCE9XX Programmer GUI)* CDCE9XXX_PROGRAMMER_GUI_v1.1.zip
   LINK: https://circuit-board.de/forum/inde...p/?s=931901dba62166cf9de532e05e8179967696b698
5. *(ClockPro™ Hex File for PSX MFO Mod)* MFO_PSX_PAL_NTSC.zip
   LINK: https://immerhax.com/wp-content/uploads/2019/03/MFO_PSX_PAL_NTSC.zip

Later today at 8:29 PM — Thread starter — #2

P

provato
Registered
Registered

28/08/20

Dear e² Journal,

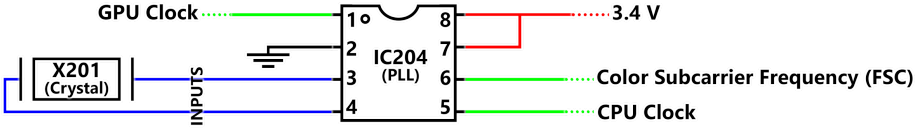
_Today I thought it's a good idea to look a little deeper into (a) the frequencies generated normally in my PAL PS1 (SCPH-7502, PU-22 board) as well as the NTSC-J and NTSC-U same board revision models (SCPH-7500 & SCPH-7501), and (b) the theory of operation of the **M**ultiple **F**requency **O**scillator (**MFO**), since I'm waiting for the

necessary parts to arrive in order to build and install this mod.

(A) Frequencies/Clocks Generated in SCPH-750x PS1s (PU-22 PCB)

_In the center of frequency generation in these PS1s lies the **IC204** chip, which is an 8-pin **PLL** (**P**hase-**L**ocked **L**oop Synthesizer).
Let's have a look at the PLL's pins (part #: Cypress CY20815L-50xx):

→ Picture 1: IC204 (PLL) Connections



→ IC204 INPUTS:

• **Pins 3 & 4:** Connected to **X201** Crystal. PAL PS1s have a 17.734475 MHz Crystal (17.73 for short) and NTSC PS1s have a 14.31818 MHz Crystal (14.32 for short). Hmmm... here is the first difference in frequencies between PAL & NTSC.

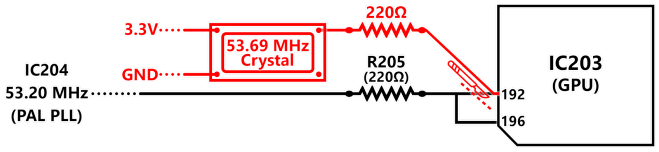
→ IC204 OUTPUTS:

• **Pin 5:** CPU Clock/Frequency. This pin is connected through a 22Ω resistor (**R204**) to CPU pin 4. The important thing to remember about this output is that it is exactly the same in both PAL & NTSC consoles (~67.73 MHz).
• **Pin 6: C**olor **S**ubcarrier **F**requency (**FSC**). This pin is connected through a 2200Ω resistor (**R501**) to **IC502** (**DAC**) pin 15, which is the **SCIN** pin (**C**olour **S**ubcarrier **IN**put). The two different frequencies outputed from pin 6 (one for PAL and one for NTSC) comply with the PAL and the NTSC Television Standards accordingly. PAL consoles generate a 4.43361875 MHz FSC (or 4.43 MHz for short) and NTSC consoles generate a 3.579545 MHz FSC (or 3.58 MHz for short). Now the selection of different X201 Crystal frequencies in PAL & NTSC makes sense: 17.734475 / 4 = 4.43361875 for PAL and 14.31818 / 4 = 3.579545 for NTSC.
• **Pin 1:** GPU Clock/Frequency. This pin is connected through a 220Ω resistor (**R205**) to GPU Pins 192 & 196. The output frequency of this pin in PAL PS1s is 53.203425 MHz which is 12 x PAL FSC (or 53.20 for short) and in NTSC PS1s is 53.693175 MHz which is 15 x NTSC FSC (or 53.69 for short). It becomes clear now how the IC204 PLL takes the X201 Crystal's inputs and creates the appropriate frequencies in each region by simple multiplication and/or division:
-NTSC GPU Clock = (15/4) x 14.32 MHz Crystal clock
-PAL GPU Clock = (12/4) x 17.73 MHz Crystal clock
-NTSC FSC = (1/4) x 14.32 MHz Crystal clock
-PAL FSC = (1/4) x 17.73 MHz Crystal clock

* **BONUS INFO:** Both PAL & NTSC SCPH-750x PS1s use the

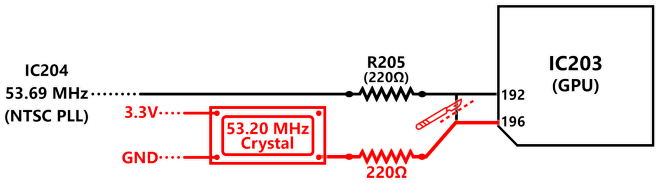
same IC203 which is the GPU chip SONY CXD8561Q. This GPU's pin 196 is utilised as the PAL video clock input when PAL video is detected. On the other hand, pin 192 is utilised as the NTSC video clock input when NTSC video is detected. These two pins are connected (shorted) together at the bottom of the PU-22 pcb. So, if someone only wants to have both NTSC & PAL correct GPU clocks in a SCPH-750x PS1 and doesn't care about subcarrier (FSC) frequencies, doesn't need a Dual or Multiple Frequency Oscillator (DFO or MFO). He only needs two things: an appropriate crystal/clock generator and a 220Ω resistor!
_For PAL consoles: Lift pin 192 (or cut the trace going to this pin) and solder the 220Ω resistor in series with a 53.69 Crystal/Clock on pin 192.

→ Picture 2: "Second Oscillator" Mod in PAL Consoles



_For NTSC consoles: Lift pin 196 (or cut the trace linking pin 196 to pin 192 on the bottom) and solder the 220Ω resistor in series with a 53.20 Crystal/Clock on pin 196.

→ Picture 3: "Second Oscillator" Mod in NTSC Consoles



_RGB connections to monitors (through SCART or BNC cables) "don't care" about color subcarrier frequency (FSC), so if someone only wants to use RGB, this simpler mod would suffice (...but I know myself, and it bugs me to not have the correct FSC in both PAL & NTSC video for possible RF, composite or s-video connections of my PS1 to various TV sets).

→ IC204 POWER CONNECTIONS:

- **Pins 7 & 8:** 3.4V (VDD)
- **Pin 2:** Ground (VSS)

_...so now that we know what the IC204 chip does in both PAL & NTSC consoles, let's summarize all the generated frequencies in a handy reference table.

→ Table 1: SCPH-750x PS1 PLL Frequencies

Name	Type	X201	FSC	Amour

		Multiplier	Multiplier	(MHz)
Color Subcarrier Frequency (FSC)	Output	PAL → x 1/4 NTSC → x 1/4	(x1)	PAL: 4.43361875 (~4.43) NTSC: 3.579545 (~3.58)
Crystal Frequency (X201)	Input	(x1)	PAL → x 4 NTSC → x 4	PAL: 17.734475 (~17.73) NTSC: 14.31818 (~14.32)
GPU Clock	Output	PAL → x 12/4 NTSC → x 15/4	PAL → x 12 NTSC → x 15	PAL: 53.203425 (~53.20) NTSC: 53.693175 (~53.69)
CPU Clock	Output	N/A	N/A	PAL → ~67.73 NTSC → ~67.73

(B) Multiple Frequency Oscillator (MFO) Theory of Operation

_Let's say we want to make a device that is able to generate ALL the required frequencies inside SCPH-750x PS1s, NTSC or PAL, and also has the ability to auto-detect NTSC or PAL games and auto-select either NTSC or PAL frequencies according to the auto-detection.

_What do WE NEED this device to do exactly?

_(i) output both GPU Clocks (PAL 53.20 MHz & NTSC 53.69 MHz).

_(ii) output both Color Subcarrier Frequencies (PAL 4.43 MHz & NTSC 3.58 MHz).

_(iii) autodetect NTSC or PAL games and then autoselect the correct frequency pair accordingly: PAL GPU/FSC Clocks pair (53.20 MHz & 4.43 MHz) or NTSC GPU/FSC Clocks pair (53.69 MHz & 3.58 MHz).

_What WE DO NOT NEED this device to do exactly?

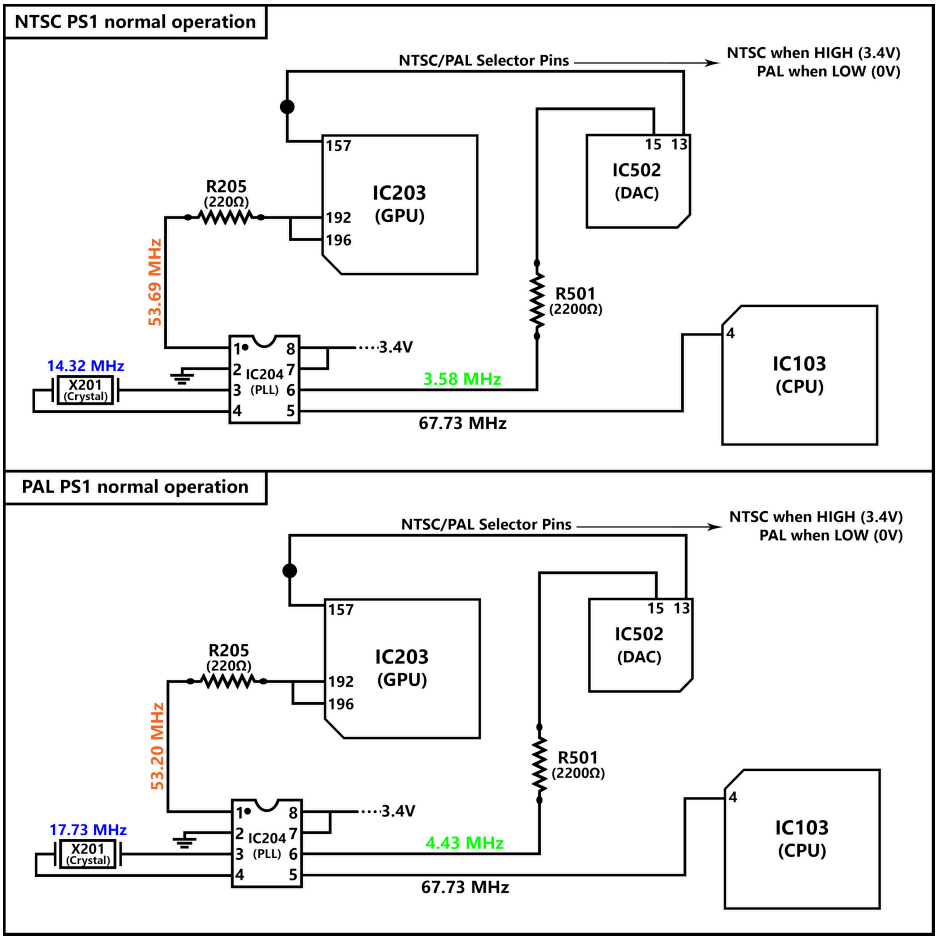
_(i) output CPU Clock, since the PLL already does this job and outputs the same exact frequency in both NTSC & PAL consoles (~67.73 MHz).

_(ii) change/remove the X201 crystal (input frequency), since the crystal does not affect the output frequencies once our device is installed.

_...Turns out this device already exists and it's called the "Multiple

Frequency Oscillator" (MFO for short) thanks to the work of two people: User "Micro" and user "skum" or "immerhax". *THANK YOU GUYS!!!*
_Let's take a step further and compare the MFO's operation with both PAL & NTSC default PLL operation.

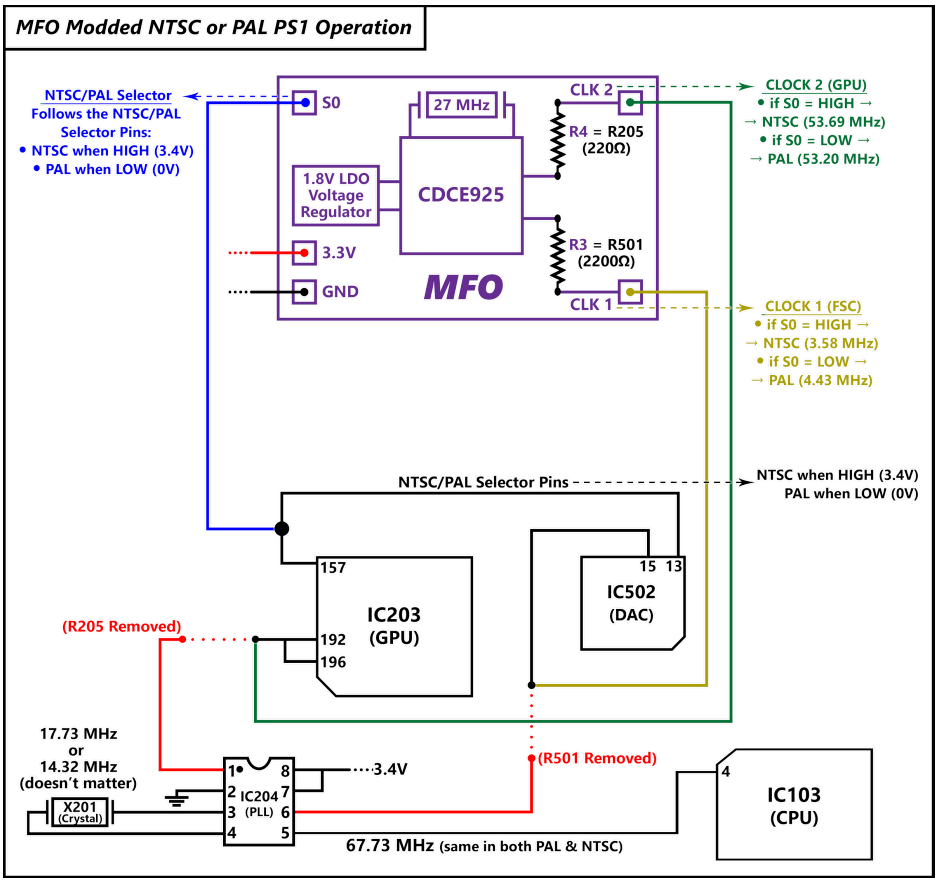
→ *Picture 4: NTSC & PAL PS1 Normal PLL Operation*



_The "heart" of the MFO is the Texas Instruments **CDCE925** clock generation IC. It must be programmed to output all four GPU & FSC Clocks in two CLK1/CLK2 pairs: CLK1=4.43 / CLK2=53.20 for PAL and CLK1=3.58 / CLK2=53.69 for NTSC. The MFO also features a "S0" selector input to autoselect the PAL or NTSC frequency pairs accordingly. Turns out all SCPH-750x PS1s also have an NTSC/PAL autodetect and autoselect feature: when GPU pin 157 and DAC pin 13 go high (3.4V) NTSC is selected, while when these pins go low (0V) PAL is selected. Thus, if S0 is connected to these pins, it will follow the inherent NTSC/PAL detection & selection (*Note #1: GPU pin 157 and DAC pin 13 are internally connected and there is a handy solder pad on this trace between them to wire to the S0 input of the MFO). Looking at the normal operation diagrams above, it seems that there are only two more things to do for the MFO to work:
-Disconnect the PLL's (IC204) GPU Clock and Subcarrier Frequency connections (Pins 1 and 6). This can be easily accomplished by simply removing resistors R205 & R501.

-Connect the MFO's CLK1 output through the freshly removed R501 resistor (2200Ω) to DAC pin 15 and CLK2 output through the (also freshly) removed R205 resistor (220Ω) to GPU pins 192/196. (*Note #2: Immerhax has it backwards on his blog/website... He has programmed his ClockPro™ .hex file to output FSC in CLK1 and GPU frequency in CLK2 even though he writes it the other way around over there...)
...oh... and don't forget to give 3.3V and Ground to the poor MFO pcb. Notice that the original PLL is still needed to generate the CPU clock in pin 5 (~67.73 in both PAL & NTSC PS1s) and that it doesn't matter anymore what frequency the X201 crystal inputs!

→ *Picture 5: MFO Modded NTSC or PAL PS1 Operation*



_As a final thought, I'd like to examine different scenarios of "frequency modded" and unmodded PS1s:

- **Unmodded PAL PS1 playing NTSC video/games**: I'll get an NTSC 4.43 video signal which TV sets either can't handle and display only black & white, or they handle poorly (messed up black levels, borders etc). The framerate will also be ~29.70 FPS instead of real NTSC 29.97 FPS, which will cause video-audio desync if playing for too long.
- **Unmodded NTSC PS1 playing PAL video/games**: Similar problems as above or worse. I'll get a PAL 3.58 video signal with wrong framerate (not exactly 25 FPS) which almost no standa... CRT TV can display (monochrome, rolling picture etc).
- **PAL PS1 with the old/classic "PAL60" mod**: This mod

"impairs" the PS1's inherent ability to autodetect/autoselect NTSC/PAL video signals and forces NTSC to PAL-60 (pseudo-PAL-60). All this does is fixing the composite or s-video black & white NTSC picture in older PAL TVs (because it uses the 4.43 frequency for color info in NTSC signal too), and the exact same problems of the first category remain (wrong framerate, color and black levels, AR, borders etc).

- **PAL or NTSC PS1 with "Second Oscillator Mod" or DFO Mod**: Perfect solution if someone ONLY uses RGB (RGB doesn't need/use the subcarrier color frequency).
- **PAL or NTSC PS1 with MFO Mod**: Perfect solution for ALL connections!!! (RF, Composite, S-video, RGB)

(*Note #3: Of course, a PAL TV set must be compatible with real NTSC 3.58 / 60Hz video signal and vice versa for the MFO to do its magic and be the perfect the solution.)
_I'd like to close this journal entry with the words of another very important user/member of the psxdev community, Trimesh, that explain perfectly the problem which the MFO solves:
"When SONY designed the PSX GPU, it had provision for operation in both PAL and NTSC modes, and had a separate clock input for each mode. However, the retail consoles had regional lockout and were only capable of playing software from their region, and hence only in one video mode. As a result, SONY laid out the board to only accept one oscillator, and installed the correct one for the video mode the console was supplied to work in. This has the result that when you switch the console to the other mode, it produces an out of spec signal, which doesn't display properly due to the frequency error in the colour subcarrier, and which also runs about 1% slow or fast depending on what the native mode of the console was."

Trimesh 13/08/2014

Getta Robo

Last edited: Later today at 1:25 PM

Later today at 8:30 PM — Thread starter — #3

P

provato
Registered

Registered

Joined: Nov 8, 2019
Messages: 23
Reaction score: 7

23/09/20

Dear e² Journal,

_At last! All the parts for the MFO from mouser, oshpark and ebay are here! The plan is simple but the journey is long and consists of 3 steps: (1) Build & Program the CDCE9XX programmers (2) Build & Program the MFO boards, and (3) Install the MFO board(s) in my PS1(s) & Test all region discs.

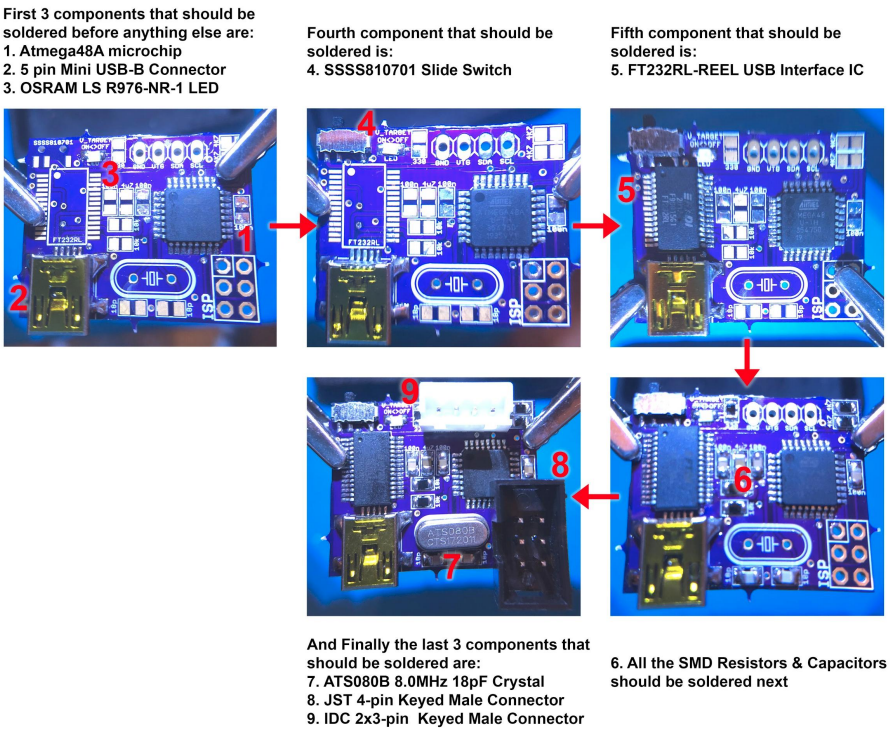
❶ Building & Programming the CDCE9XX Programmers

1a. Sodering Parts to the CDCE9XX Programmer Board

_Seeing the PCBs it's obvious I first have to cut the connector edge/ tip at the dashed line. I open up a new & fresh No. 11 surgical scalpel and carefully carve a straight line right on the dashed line on both sides. The more cuts - the better. Man..! these boards are tough to cut..!

_After that it is time to fire up the soldering/desoldering station and solder the components one by one (of course I could add a tiny bit of solder on the pads and then bake the pcb with the hot air station, but nah... where's the fun in that? - maybe another day). On these little pcbs it's important to know where to begin and where to end, because if I install parts randomly, one part might present as an obstacle in the soldering of the next. Another thing to watch out for is the polarity of things. In this particular board I chose the keys of the connectors to point inside (for easy remembering) and I checked the polarity of the LED with my DMM before installing it. Of course the cathode (negative) goes to ground.

→ *Picture 1: Soldering on the CDCE9XX Programmer*



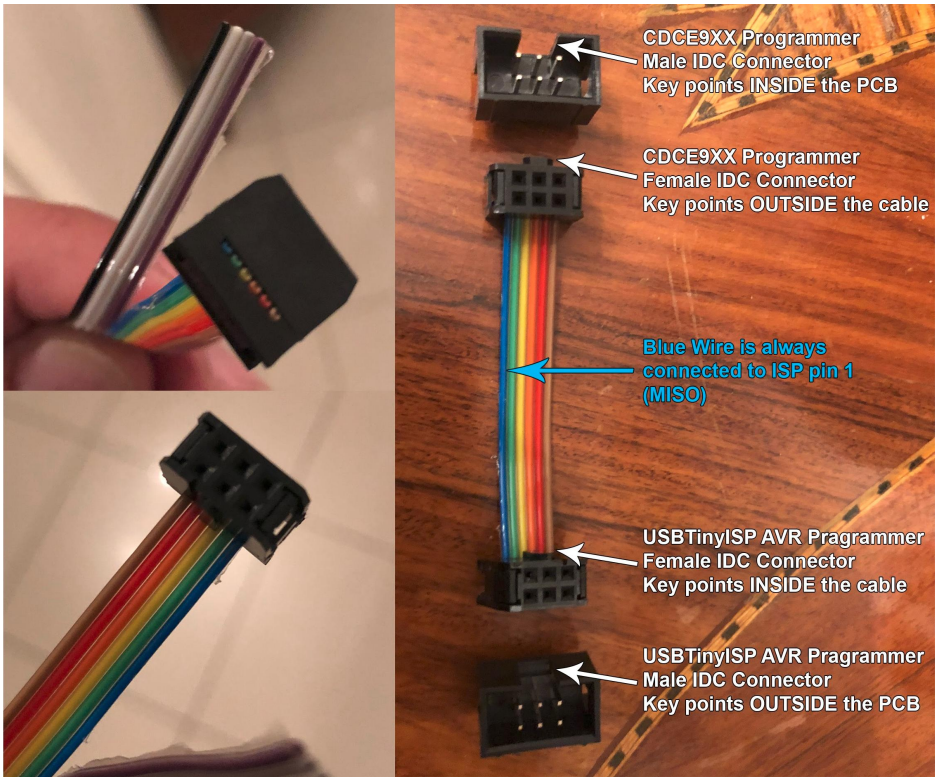
1b. Making the Cables & Connectors for the USBtinyISP AVR Programmer and the MFO PCBs

_Next up is cables and attaching connectors to them (I could of course just solder the cables on the pcbs, but I LOVE connectors!). One thing to point out is that it's really nice when little things in life add up in the end... What I mean by that is that the ebayer sent me as a present a 1m long 10-colour cable and I need a 6-colour cable for the ISP connection plus a 4-colour one for the MFO (I2C) connection. This means 3 equal vertical cuts and 1 split on this cable (6/4) and it's done! I have all the cables I need!

_I went with the easy ISP cables+connectors first this time - just

pressed the top of the 2x3-pin female IDC connector WITH a flat surface ON a flat surface and it's ready. However, since the female connectors are also keyed to fit their male counterparts, caution is needed with the keys' orientation. I chose to have one female connector pointing outside and the other inside the cable. This creates two situations: (a) blue wire is always MISO (rectangle) and I cannot fit the cable backwards, and (b) the male IDC connector key must point OUTSIDE on the USBtinyISP (since the male CDCE9XX programmer IDC connector key points INSIDE).

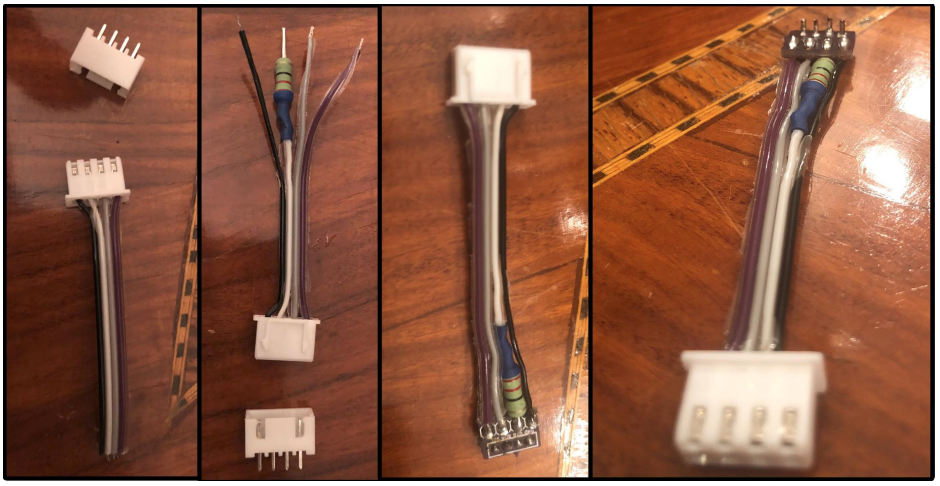
→ Picture 2: CDCE9XX Programmer ISP Cable & Connectors



_After the somewhat easy ISP cable, it's time to make the cables that connect the MFO boards to the CDCE9XX programmer (I2C cable). Here is where things get a little tricky, and I have to be triple careful. First, the programmer side of the 4-wire cable must have a JST female connector with the key pointing OUTSIDE (UP) the cable, since its counterpart JST male connector on the CDCE9XX programmer board points INSIDE. A wire stripper and a wire crimper get this job done very easily. Second, the VTG (+ voltage) white wire, which is the second from left, must be connected to the corresponding VTG connection on the MFO board via a 22Ω resistor to limit the current it draws the moment I connect the two boards until the capacitors get charged (too much current in the beginning is known to cause resets to the programmer!!!). Third, the connector on the MFO side has a 2mm spacing instead of the 2.54mm spacing of all the connectors I have... This is where I have to use the connector edge/tip I cut earlier off the CDCE9XX programmer pcb. This tip will serve as a connector to the MFO, but it's also the most difficult part to make in order to have a snap-on / snap-off connection to the MFO. If someone

watches this little edge pcb closely, it is obvious that it is in fact a 2.54mm to 2mm spacing female-female adapter. The plan is to solder the 4-wire cable on the 2.54mm side, while on the 2mm side I'll solder four ~5mm long cut connector pins vertically, that will plug directly on an "improvised" female connector on the MFO. More on that improvisation later when I get to the second step of the whole MFO process...
_The following picture shows the correct orientation of this cable's connectors' both pins & keys.

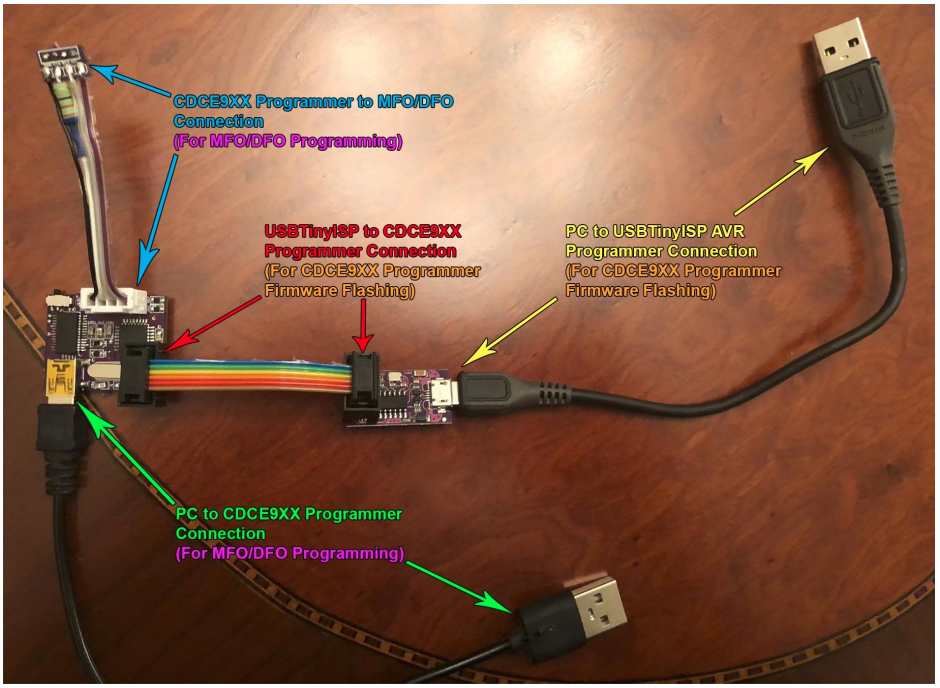
→ Picture 3: CDCE9XX Programmer I2C Cable & Connectors



_Last but not least, two more cables are needed for PC connections: A USB-A male to USB-B Mini male cable (for PC to CDCE9XX programmer connection) and a USB-A male to USB-B Micro male cable (for PC to USBtinyISP AVR Programmer connection).

→ Picture 4: All CDCE9XX Programmers and their Cables

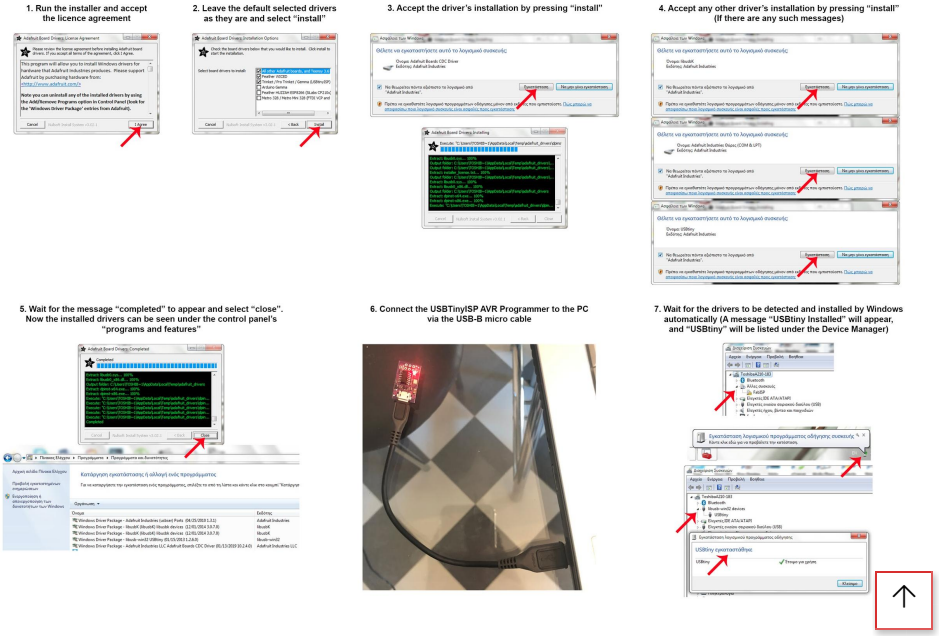
→ Picture 5: All CDCE9XX Programmer Connections



1c. Flashing the firmware and setting the fuse bytes on the CDCE9XX programmer

_All the necessary hardware for the CDCE9XX programmer is now ready, so it's time I move on to the software. The USBtinyISP AVR programmer, like any other USB device, needs its drivers and a windows application (GUI) in order to operate when connected to a PC. Thanks to user "bmp02" on nfggames.com forums I know which windows drivers (adafruit USBtinyISP drivers - got it from https://learn.adafruit.com/usbtinyisp/drivers) and which windows GUI (AVRDUDESS - got it from http://blog.zakkemble.net/avrdudess-a-gui-for-avrdude/) to download and install. Installing both of these is easy and there are online guides for windows OS on the mentioned websites, but anyway, here are my guides too:

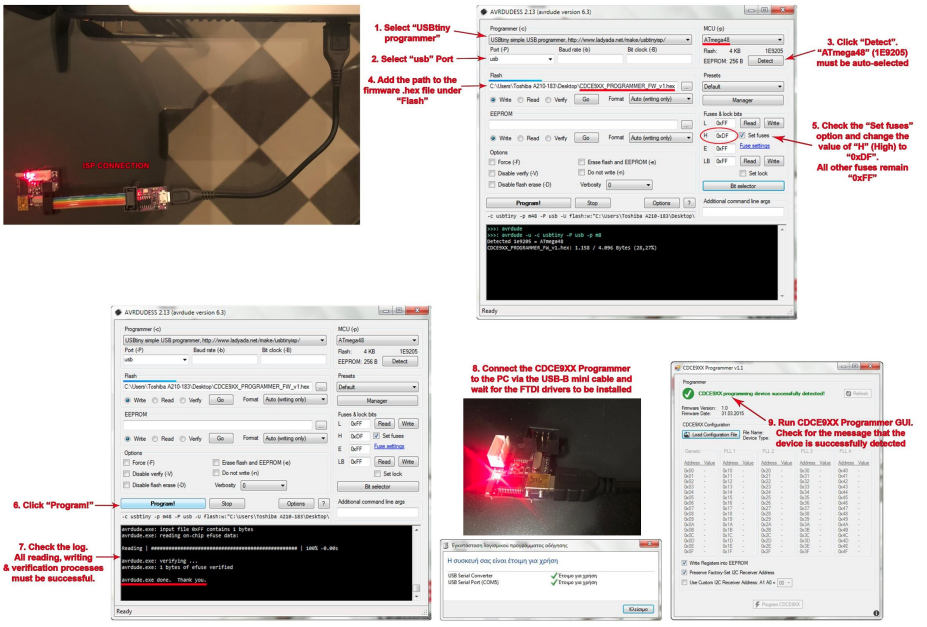
→ *Picture 6: Installing the USBtinyISP AVR Programmer Drivers*



→ *Picture 7: Installing the AVRDUDESS GUI*

_Next, even though it is not necessary to install windows drivers for the USB to Serial UART chip on the CDCE9XX programmer (because Windows automatically detects & installs the correct-current drivers), I went ahead and downloaded/installed them from the official FTDI website (https://ftdichip.com/drivers/D2XX.htm), (file: CDM21228_Setup.zip)
_It's time now to run AVRDUDESS and flash the firmware - set the fuse bytes to the Atmega48A chip on the CDCE9XX programmer. The USBtinyISP AVR programmer must be connected to the PC with the appropriate USB cable AND to the CDCE9XX programmer with the ISP cable (*Note #1: It doesn't matter if the CDCE9XX programmer is turned ON or OFF). In AVRDUDESS, the "usbtiny simple usb programmer" under "Programmer (-c)" and "usb" under "Port (-P)" must be selected. After that, the "Atmega48" must be detected automatically when "Detect" is pressed. Next, under "Flash" the .hex file which is the CDCE9XX programmer's firmware must be selected. Lastly, the "Set fuses" checkbox must be checked and all fuses must have a value of "0xFF" except for "H" (High Fuse) that has to be changed to "0xDF". By clicking "Program!" after all these settings, the Atmega48A chip should be successfully programmed and ready to use for DFO/MFO programming.
_The AVRDUDESS log at the bottom must show that all writing, reading & verification processes of the the firmware (flash) and the fuse bytes are successful, and display an "avrdude.exe done" message in the end. If an error message is displayed instead, closing AVRDUDESS, disconnecting all cables and then repeating the whole process again might solve the problem. If not, most likely something went wrong with soldering parts or cable making or loose connections.
_The best way to test that the CDCE9XX programmer was programmed correctly, is to connect it to the PC with the USB-B Mini cable, wait for Windows to install FTDI drivers and then run its own GUI. If it says "...successfully detected!", the CDCE9XX programmer is ready to program DFO/MFO boards (*Note #2: Again, it doesn't matter if the CDCE9XX programmer is turned ON or OFF).

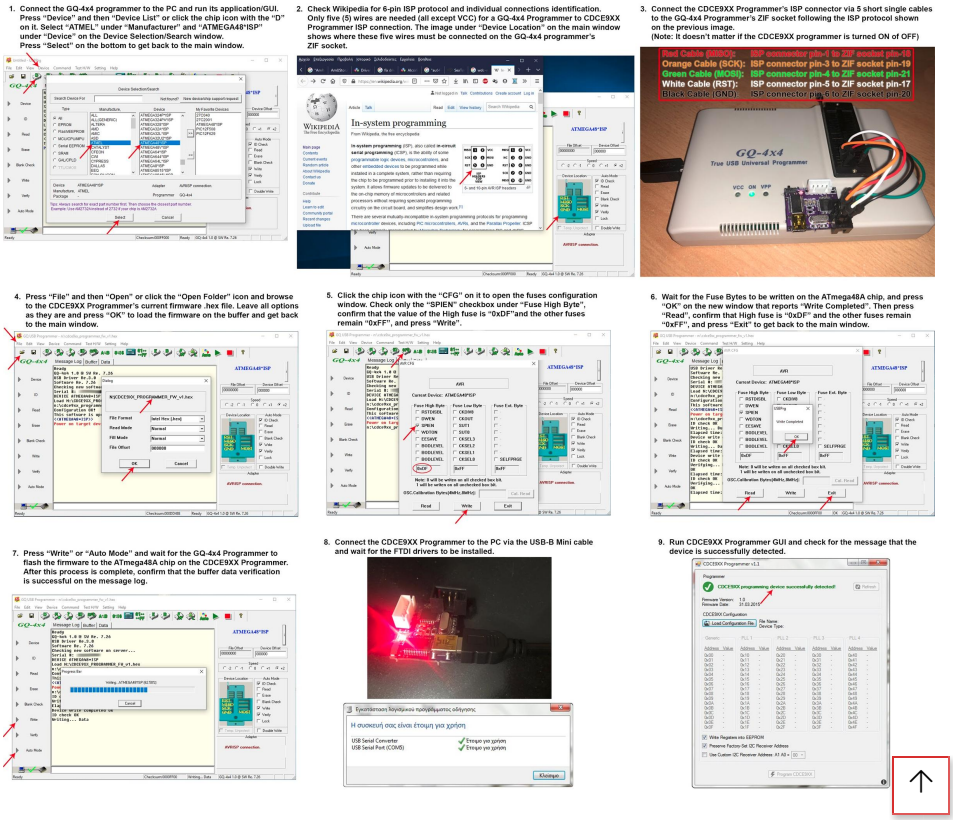
→ *Picture 8: Programming the CDCE9XX Programmer with the USBTinyISP AVR Programmer*



1c (Alternative). Flashing the firmware and setting the fuse bytes on the CDCE9XX programmer with the GQ-4x4 universal programmer

_As it turns out, I spent extra money and time on USBtinyISP unnecessarily, since my GQ-4x4 Universal Programmer can very well program through ISP the Atmega48A chip (and all other Atmel chips). The following picture describes the GQ-4x4 ISP procedure for the CDCE9XX programmer.

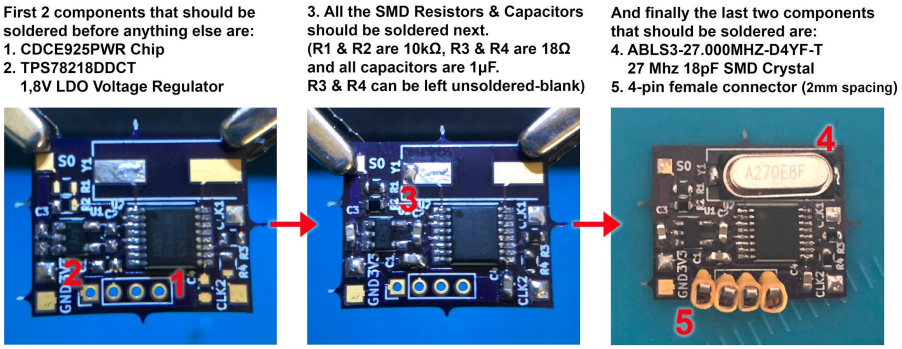
→ *Picture 9: Programming the CDCE9XX Programmer with the GQ-4x4 Universal Programmer*



❷ Building & Programming the MFO Boards

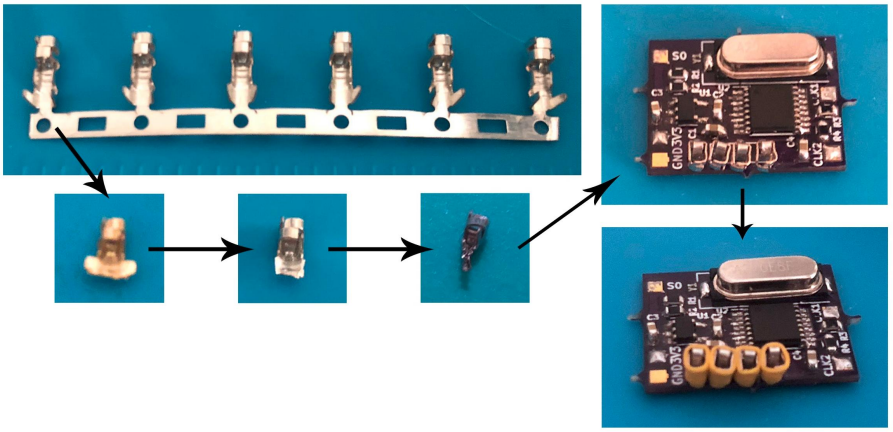
_Everything went smoothly up until this point, so I'll deal with the end product - the MFO boards - now. As with the CDCE9XX programmer boards, the tiny size of the MFO dictates for a correct order of soldering parts on it, which is shown in the following picture:

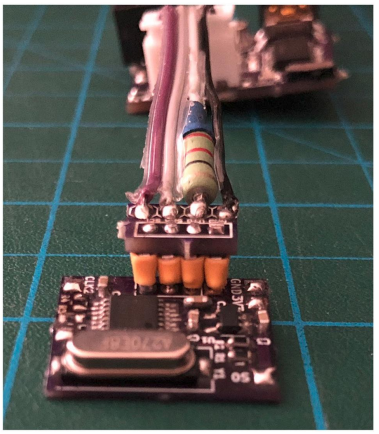
→ *Picture 10: Soldering on the MFO*

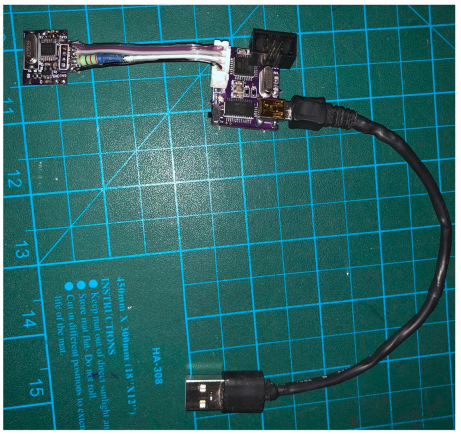


_As I mentioned earlier, I was a fool not to order additional JST or IDC connectors with 2mm spacing (pitch), so the male connector on the MFO side of the I2C cable and the female connector on the MFO board had to be "improvised" (of my own device). For the connector on the MFO board, I cut out four JST standard female single connectors, then trimmed off their lower two cable hooks and rolled the remaining two upper cable hooks until they formed into a "pin" that would fit in the 2mm-pitch holes on the MFO pcb. Then I lined up the four modified single connectors on the four holes on the MFO, soldered them and wrapped them in heat-shrink tubing to prevent them from shorting one another. It worked like a charm!

→ *Picture 11: Improvised Female 2mm Pitch MFO (I2C) Connector*

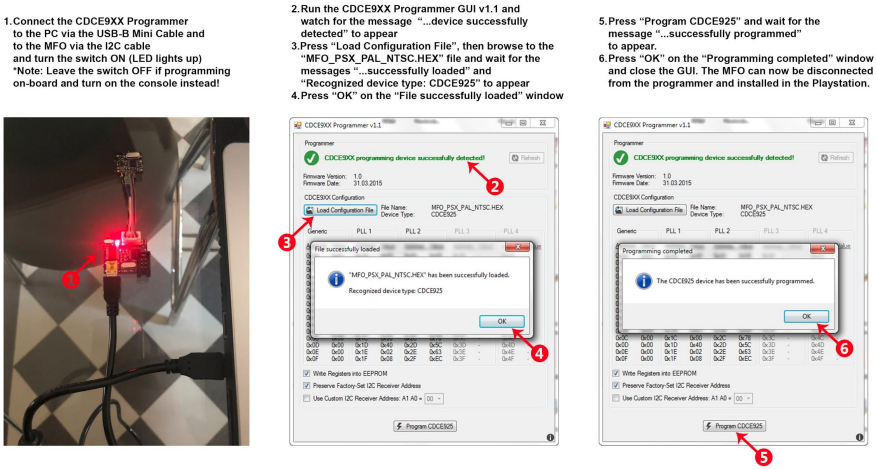






_Programming the MFO boards (I mean the CDCE925 chips on them...) is a very easy task thanks to the work of user "micro" (inventor of the DFO) who provided the necessary software for the job (CDCE9XX Programmer GUI v1.1) - easy as pressing two buttons!:
_(i) Connect the CDCE9XX programmer to a Windows PC via the USB-B Mini cable and to the MFO with the I2C cable. Now the programmer's switch must be turned ON!
_(ii) Run the CDCE9XX Programmer GUI v1.1 and watch for the message "...successfully detected!" to appear, then press "Load Configuration File", browse to the "MFO_PSX_PAL_NTSC.HEX" file (provided by user "skum" / "immerhax") and wait for the messages "...successfully loaded" & "Recognized device type: CDCE925" to appear.
_(iii) Finally press "Program CDC3925" and wait for the message "The CDCE925 device has been successfully programmed" to appear.
_The MFO pcb is now ready to be installed inside the PS1 after that.

→ *Picture 12: Programming the MFO*



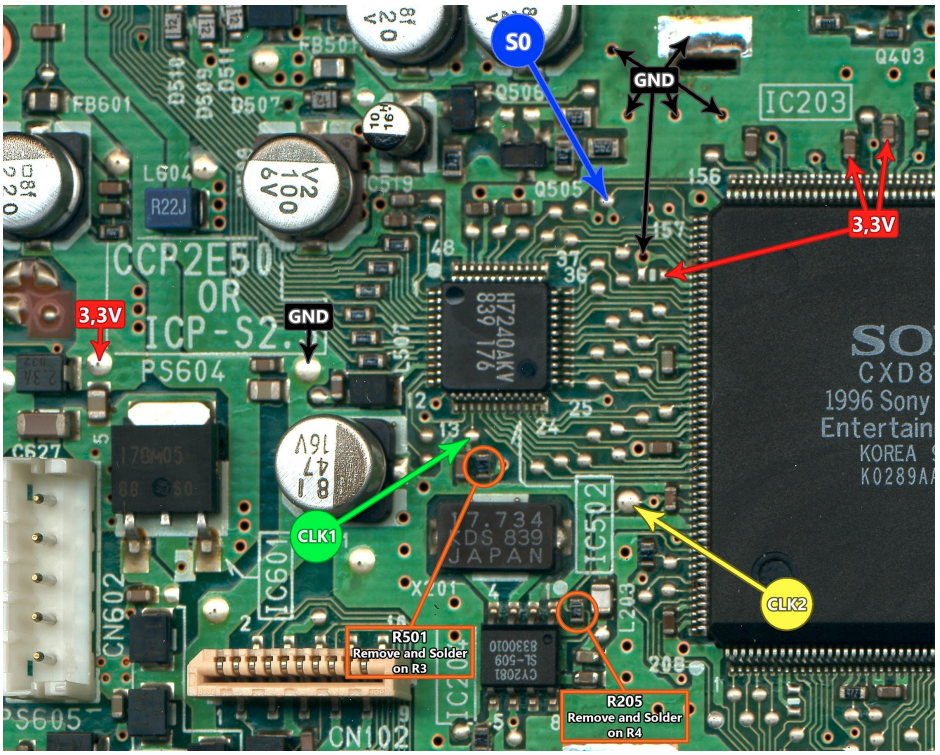
_(*Note #3: The MFO pcb can be programmed even if it is installed in the playstation, but CAUTION is needed with this operation not to give the poor pcb more - twice as much - voltage than it needs and burn it. As micro describes this issue: *"It's important to set the V_Target switch on the programmer hardware right. If you have the DFO sitting on your table and you want to program it, set V_Target to on. The L should start to shine red. If you want to program a DFO that's already*

installed in a system (let's say in a Megadrive for example), then turn V_Target off. Instead turn on the Megadrive itself. It will power the DFO. If you mess this up you can damage your console, the programmer and even your PC!")

❸ Installing & Testing the MFO boards

_Ufff... last step before I enjoy an all-region faithful video signal on my PSX. By now it should be obvious what has to be soldered/unsoldered on the MFO and the PU-22 pcb for the mod to work:
(a) unsolder resistors R501 and R205 off the PU-22 board and solder them on R3 and R4 respectively.
(b) Solder 5 wires from the MFO to the PU-22 pcb (**S0** to the pad going to GPU pin 157 & DAC pin 13, **CLK1** to the pad going to DAC pin 15, **CLK2** to the pad going to GPU pins 192 & 196, **3.3V** to a pad that provides this voltage and **GND** to a ground pad).

→ Picture 13: MFO Installation Diagram (PU-22 Board)



→ Picture 14: MFO Installation Examples (PU-22 Board)

Later today at 8:30 PM

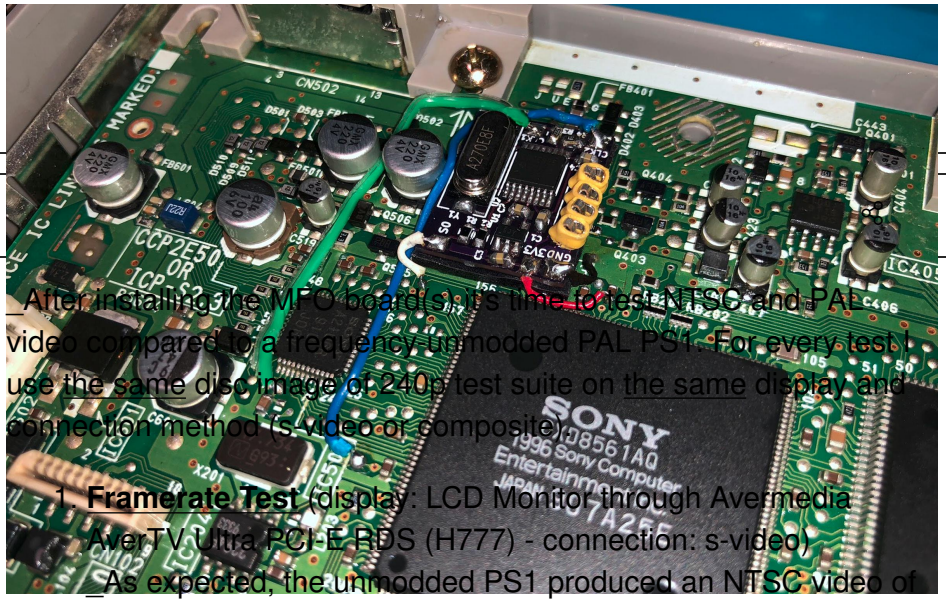
P

provato

Registered

Registered

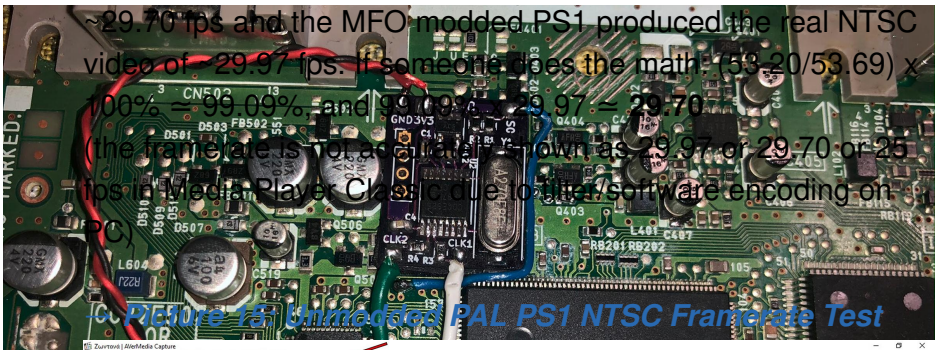
Joined:	Nov 8, 2019
Messages:	23
Reaction score:	7



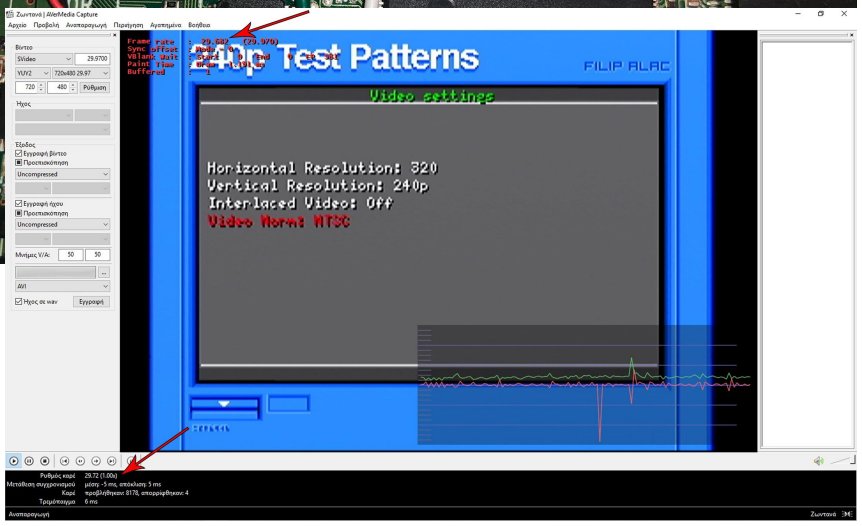
After installing the MFO board(s), it's time to test NTSC and PAL video compared to a frequency unmodded PAL PS1. For every test I use the same disc image of 240p test suite on the same display and connection method (s-video or composite).

1. **Framerate Test** (display: LCD Monitor through Avermedia AverTV Ultra PCI-E RDS (H777) - connection: s-video)

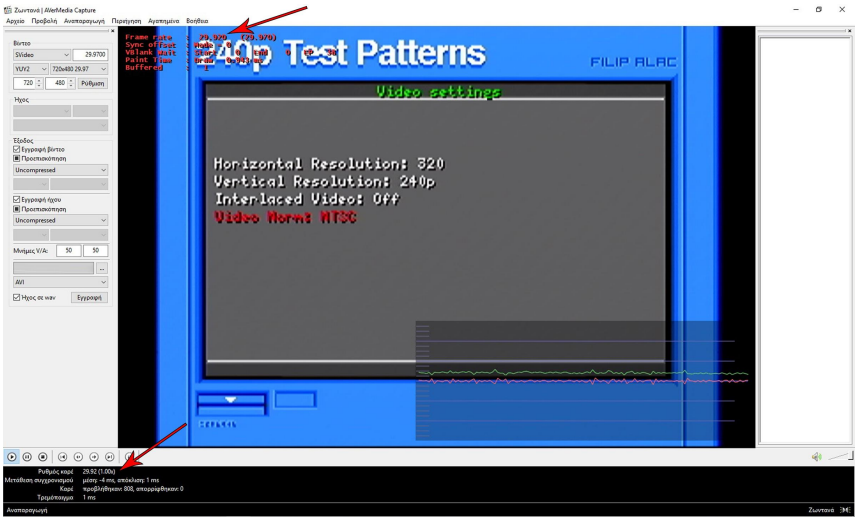
As expected, the unmodded PS1 produced an NTSC video of ~29.70 fps and the MFO modded PS1 produced the real NTSC video of ~29.97 fps. If someone does the math: (59.20/53.69) x 100% = 99.09%, and 99.09% x 30.97 = 29.70 (the framerate is not actually shown as 29.97 or 29.70 or 25 fps in Media Player Classic due to ultra/software encoding on PC)



→ *Picture 15: Unmodded PAL PS1 NTSC Framerate Test*

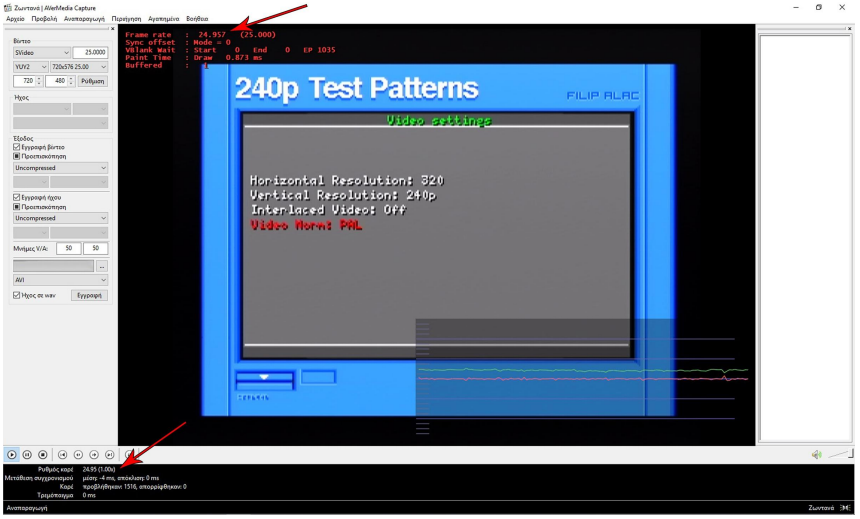


→ *Picture 16: MFO Modded PAL PS1 NTSC Framerate Test*

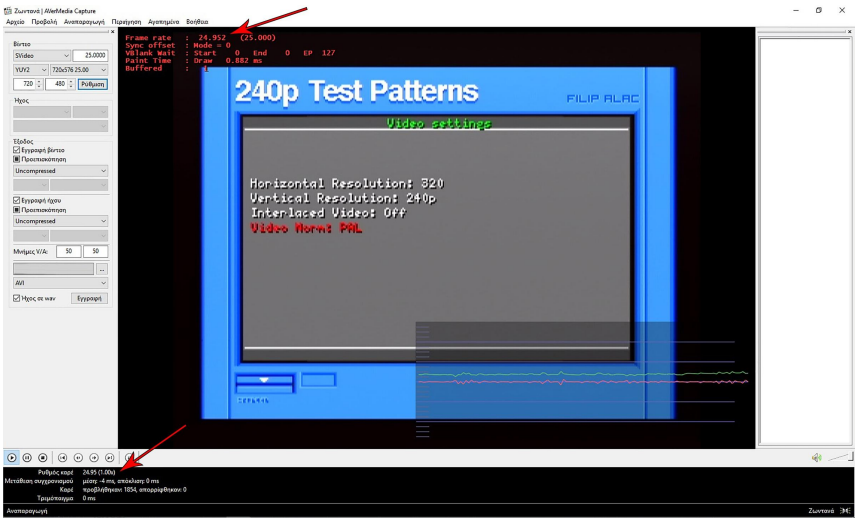


_As for PAL framerate, the MFO modded and unmodded PS1 produced the same ~25 fps PAL video.

→ *Picture 17: Unmodded PAL PS1 PAL Framerate Test*



→ *Picture 18: MFO Modded PAL PS1 PAL Framerate Test*

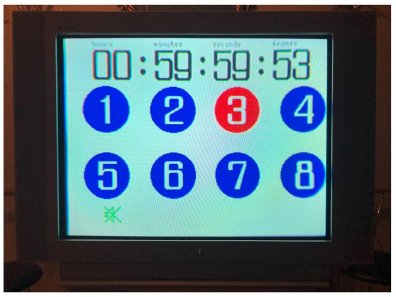


2. **NTSC Timing Error Test** (display: Sony Trinitron KV-29FX30E, connection: composite)
   _This test is designed to check if the output video of a console

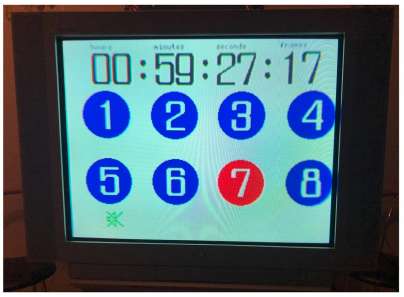
is slower or faster than it should be. I connected both of my SCPH-7502 PS1s (unmodded and MFO modded) to the CRT TV (AV1 & AV2 scart inputs through composite cables) and then loaded the 240p test suite's "Lag Test" on both consoles, after selecting NTSC video on both first of course. At the exact same moment I pressed the "X" buttons on both PS1 controllers and the timer on my iphone. The timers (2 on the CRT TV and 1 in my iphone) started running simultaneously. After exactly one hour (60 minutes or 3600 seconds), I again pressed the "X" buttons and the iphone timer simultaneously. As expected, the timer on the iphone and the MFO modded PS1 showed exactly 60 minutes, while the timer on the unmodded PS1 was 33 seconds behind/slower (59' : 27" or 3567 seconds). If someone does the math again: (53.20/53.69) x 100% ≃ 99.09%, and 99.09% x 3600 seconds ≃ **3567 seconds** or **59' : 27"**

→ Picture 19: 33 seconds per hour Timing Error in unmodded PAL PS1 displaying NTSC video

MFO Modded PAL PS1 displaying NTSC Video (Accurate per hour Timing)



Unmodded PAL PS1 displaying NTSC Video (33 Seconds per hour Timing Error)



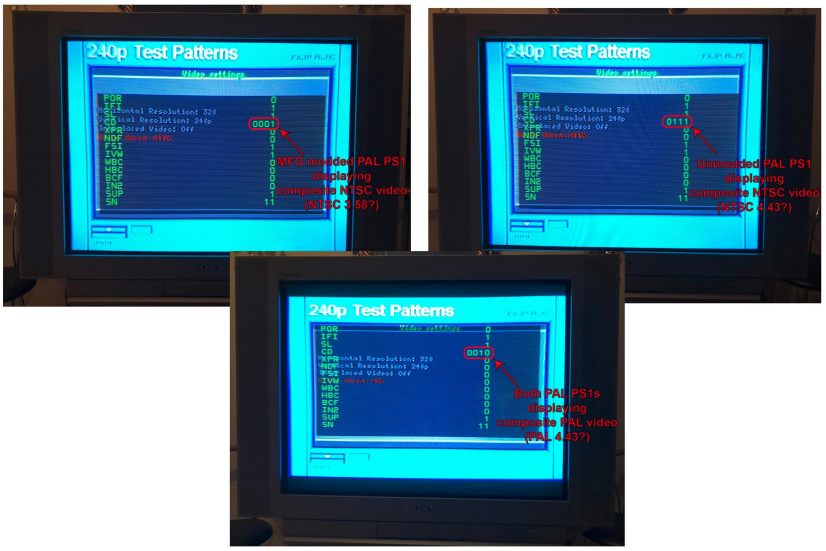
_(*Note #4: Konami's "Benami" games (e.g. beatmania or dance dance revolution) as well as other rythm games on the PS1 need accurate timing in order to be playable. Otherwise the video and audio will get desynchronised. The MFO fixes the timing when playing other region such games, so they become completely and accurately playable with this mod!)

3. **NTSC Colours, Black Levels, AR and Screen Position Test**
   _Since all my CRT TVs and my Avermedia TV card can handle NTSC 4.43 perfectly fine, I didn't notice huge differences in colours, black levels and picture geometry. NTSC video of the unmodded PAL PS1 gave slightly darker black & shifted to the right image on the AverTV, and slightly brighter blacks & shifted to the right image on the Sony Trinitron. I hear people with scalers (OSSC, framemeister etc.) notice huge improvement with the DFO, and I imagine (since I don't own such a scaler) that the MFO will produce perfectly accurate results for them when using composite or s-video.

4. **Video Standard Recognition by Sony Trinitron**
   _I spent a lot of time in my KV-29FX30E's service menus to find out if it recognizes an NTSC composite video input as NTSC

3.58 for the MFO modded PS1 and as NTSC 4.43 for the unmodded one. After entering the "status menu", I noticed under the abbreviation "CD" the 4-digit number changed for each input:
-Unmodded PAL PS1 displaying NTSC video → "CD 0111" (I guess this is NTSC 4.43?)
-MFO modded PAL PS1 displaying NTSC video → "CD 0001" (NTSC 3.58?)
-Both PAL PS1s displaying PAL video → "CD 0010" (PAL 4.43?)

→ Picture 20: Video Standard Recognition by Sony Trinitron in "status menu"



G

_Unfortunately my google skills coudn't bring me an answer as to what the "CD" abbreviation stands for in Sony Trinitron CRT TVs... I hope one day I'll find out.

THE END

(oh... did I mention that MFO & DFO boards can be used on other video game consoles too, in order to have accurate PAL 4.43 50Hz and NTSC 3.58 60Hz video? Well... these are projects for another day and another entry here..!)

About

Obscure Gamers is a Video Game Preservation group founded in 2017. We actively work to preserve long lost video game history & hardware for educational research and historical purposes. Our long term goal is to work with Video Game Developers in preserving this important history from being lost.