Journal notes on Blaupunkt Berlin IQR83 restoration

- This power point contains random information I've collected while restoring and modifying of my Blaupunkt Berlin IQR 83 car radio set.
- I've made them to organize my thinking and analysis. My work has been following a fuzzy path and that is reflected here.
- I'm sharing my journal notes in the hope that they may be useful. Information is provided without any kind of warranty. Assume my notes are wrong.
- The system is rather complex and consists of four separate units:
    - The gooseneck mounted driver interface module with the main power button, LCD display to show radio status, and operator buttons for volume and frequency control, containing a 4 bit microcontroller to interface to the main control unit
    - The under-dash (hidden) mounted radio and main control unit containing the central 16 bit micro computer system with speech synthesizer, and the FM and AM radio.
    - The dash mounted stereo auto-reverse tape deck with volume, tone, fader, and balance control buttons, and a microphone and associated electronics to enable automatic volume control depending on ambient noise in the car
    - The four channel BQB 80 booster amp connecting the system to up to four car-speakers
- The units are interconnected through shielded cables 8-pin DIN connectors carrying data and audio signals. Permanent power is fed from the car battery separately to the control unit, tape deck unit, and power amp. The operator module is powerd by the control unit through a 7-pin DIN connector.
- The system was advertised as "the worlds most expensive car radio" in 1983 when it was launched. It was preceded by the Berlin 8000 system from which the tape deck and booster amp seems to have been carried over. The driver interface and central computer, however, seems completely new for this version. My IQR 83 is the top level edition with speech synthesis and automatic station identification so that's the version I'm describing here.

Anders Dinsen
anders@dinsen.net

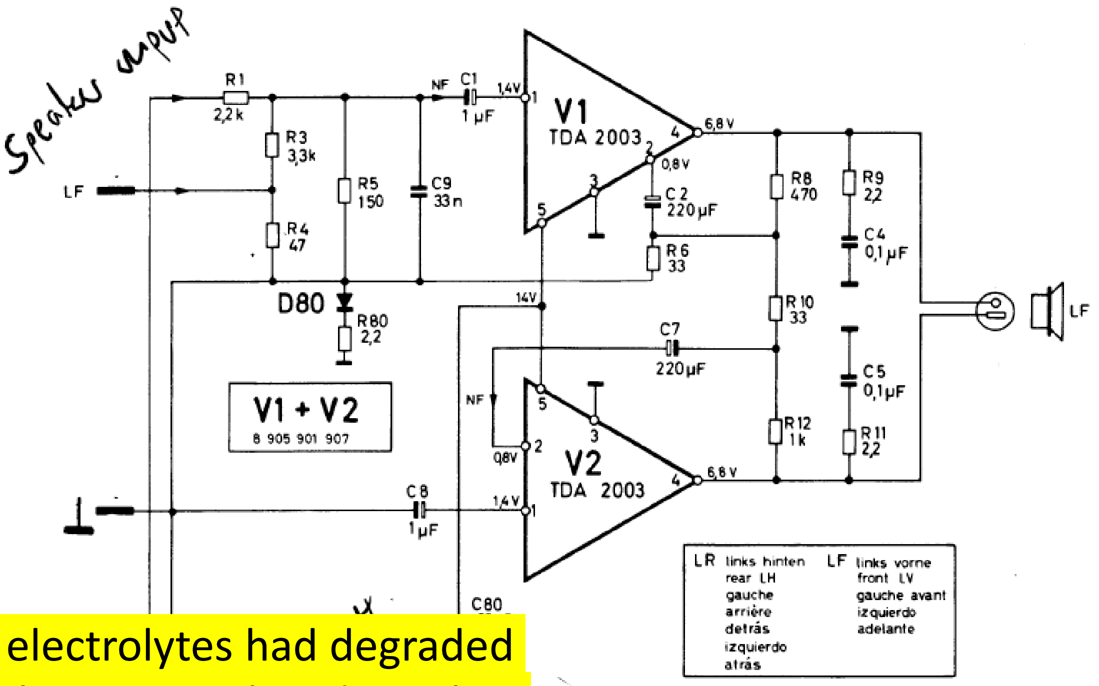
Started: 2023-10-28

BQB 80

The booster amp is based on eight TDA2003 power amplifier chips in bridge coupling two-and-two to form the four channels. Output power is specified to be 15W RMS per channel, which seems realistic, though the THD curve shown on the box indicates that 10W RMS is the most which is achievable at reasonable distortion levels. That's still a decent power level in a car by 1980's standards.

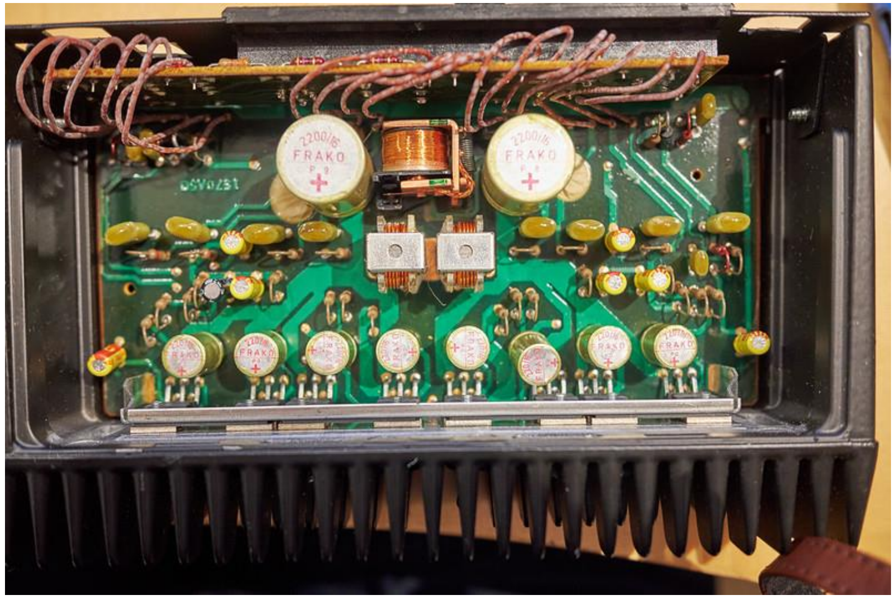
The schematic is pretty straight forward with each of the four channels modelled after the same recipe.

There are speaker inputs with 4.8 and 6.3 mm tabs and inputs from the tape deck through the 8 pin DIN connector. The speaker inputs are obviously high level, but the DIN input seems to be high level as well. A voltage divider at the input reduces the sign by -24 dB and -27 dB respectively for the DIN and speaker inputs. The voltage gain of the TDA 2003 in standard coupling is 40 dB according ot the datasheet but since they ar bridged, in total, the booster probably provides sound level amplification of 13 – 18 dB.



Update: All electrolytes have been replaced. The 2200 uF and 220 uF electrolytes had degraded significantly to 5-700 uF and ~150 uF respectively. The 1 uF seems ok, but was replaced. Cooling of TDA2003 chips has been improved by using paste between them, the mica shims and the heat sink.

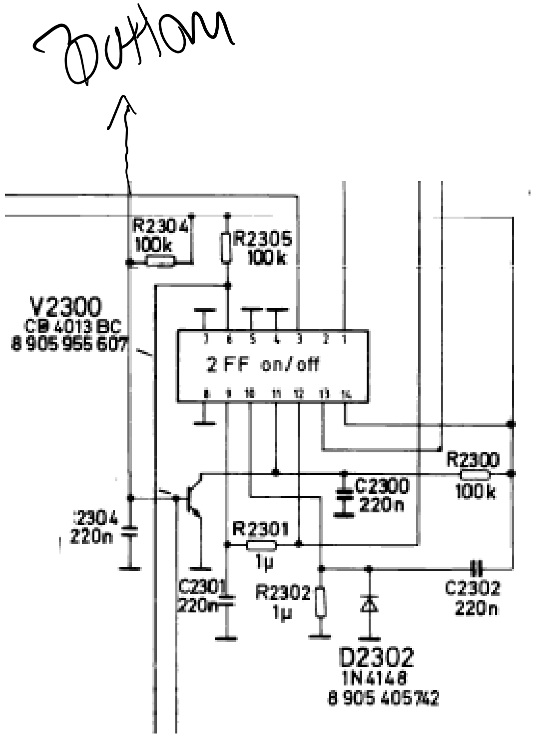
~~I'm going to use the booster with my 1989's Blaupunkt Paris until I have the Berlin working~~

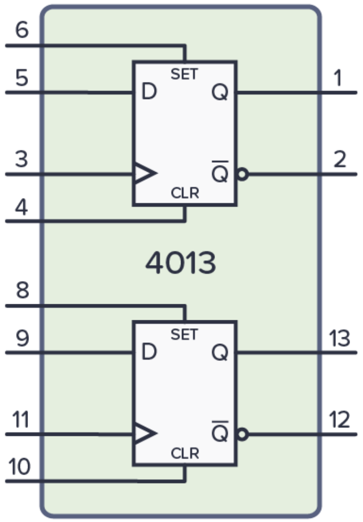


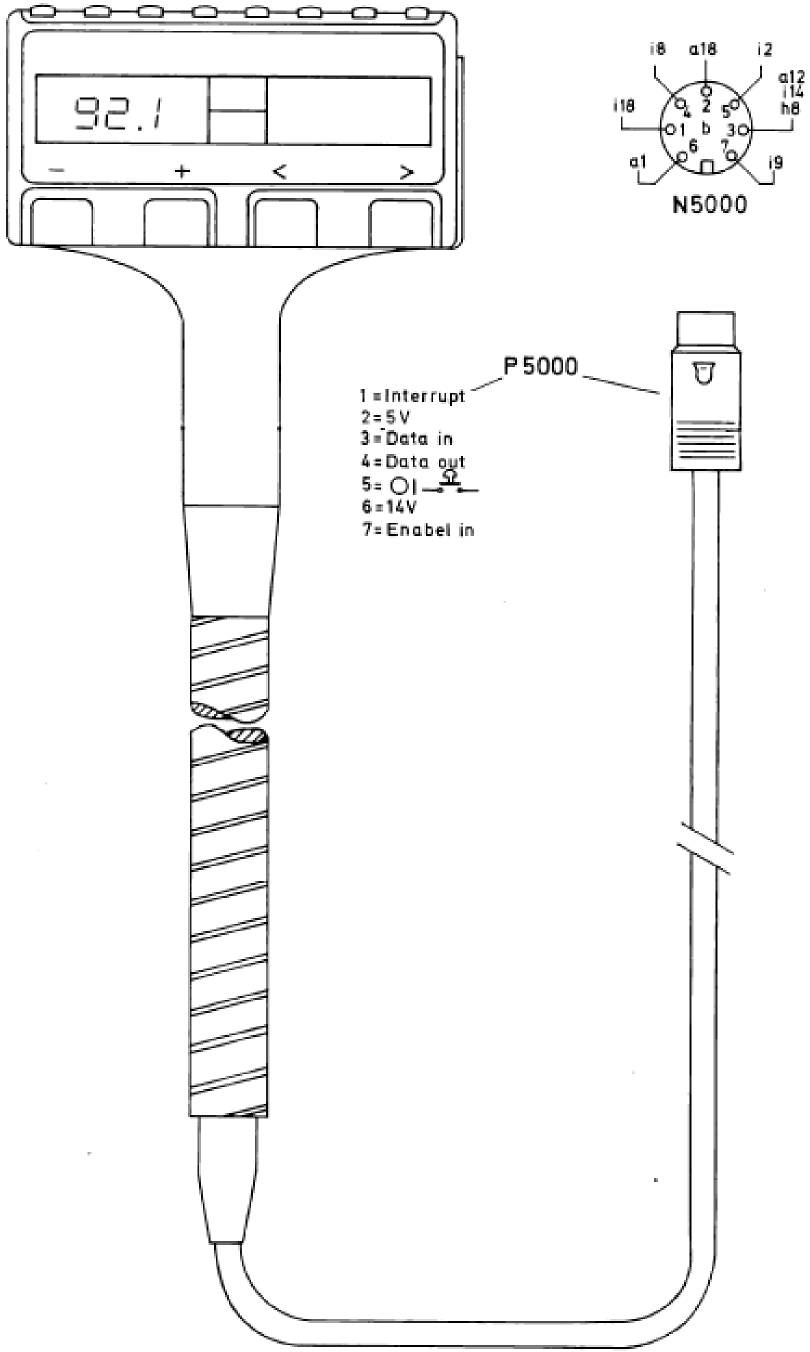
2023-10-27
2023-11-16

System Power control

- Power is controlled from the operator panel button on the right side. The button controls the positive clock input on the one half of a 4013 flip-flop that toggles the power on-state.
- The input in pin 5 on the DIN connector and is connected through a transistor with a capacitor and pullup resistor functioning as an inverter and debouncer.
- The button probably to connect to ground inside the panel although I have not been able to confirm that by measurements.







Starting main controller code analysis

The radio doesn't work and there are some electrolytes that have leaked. I therefore needed to disassemble the radio unit completely. This gave me access to the EPROMs and I tried to read the TMS2532 4k main EPROM

```
00000000: 2000 088a 20e0 041e 203e 0236 203e 022c   ... ... >.6 >.,
```

Reset workspace pointer

Reset vector

```
00000880: 3233 3441 4243 0809 0a0b 0d5b 1244 14de   234ABC.....[.D..
00000890: 111e 0000 0e6a 0e70 1188 0984 0d74 0d74   .....j.p.....t.t
```

- 0d5b = 0000 1101 0101 1011 … this is not a valid opcode. Well my disassembler can't identify a single instruction in the ROM anyway!

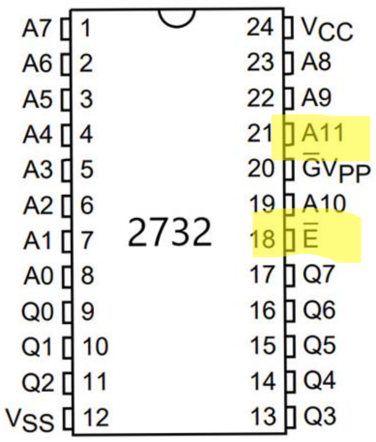
Swapping… how about:

- 5b0d = 0101 10 1100 00 1101 = SZCB indexed by r12 by r13

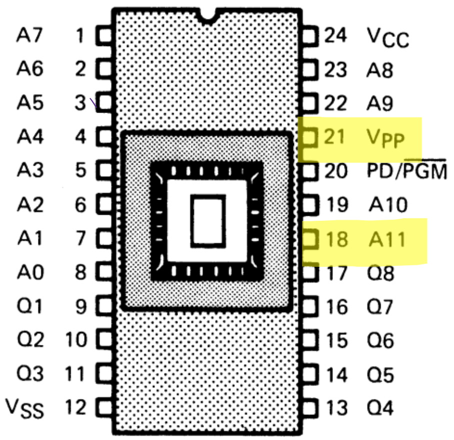
Yes that's it. The instruction bytes seem swapped for some reason, though it's an odd instruction to reset with…

This explains the odd instruction at the reset vector as it is just a random instruction at 008a, not 88a, but why are the bytes of the instructions apparantly swapped when the vectors arent? There are no hints of this behaviour in the TMS9981 datasheet…

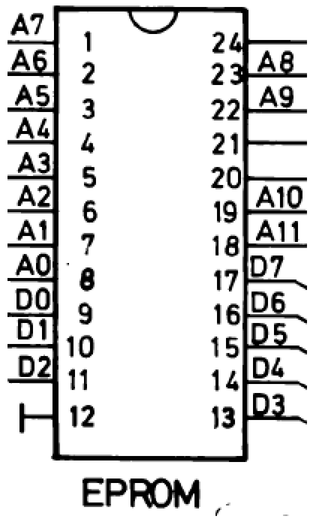
2732 Pinout



TMS2532 pinout



Blaupunkt schematic



Reading the eprom as a 2732 is only valid for the first 2048 bytes

```
00000000: 2000 088a 20e0 041e 203e 0236 203e 022c   ... ... >.6 >.,
00000010: 0016 2074 0004 0108 2050 000a 0110 208c   .. t.... P.... .

00000800: 2000 088a 20e0 041e 203e 0236 203e 022c   ... ... >.6 >.,
00000810: 0016 2074 0004 0108 2050 000a 0110 208c   .. t.... P.... .
00000820: 0002 004b c02e fffe 0240 000f 020c 8000
```

Confirmed… I need to make an adapter

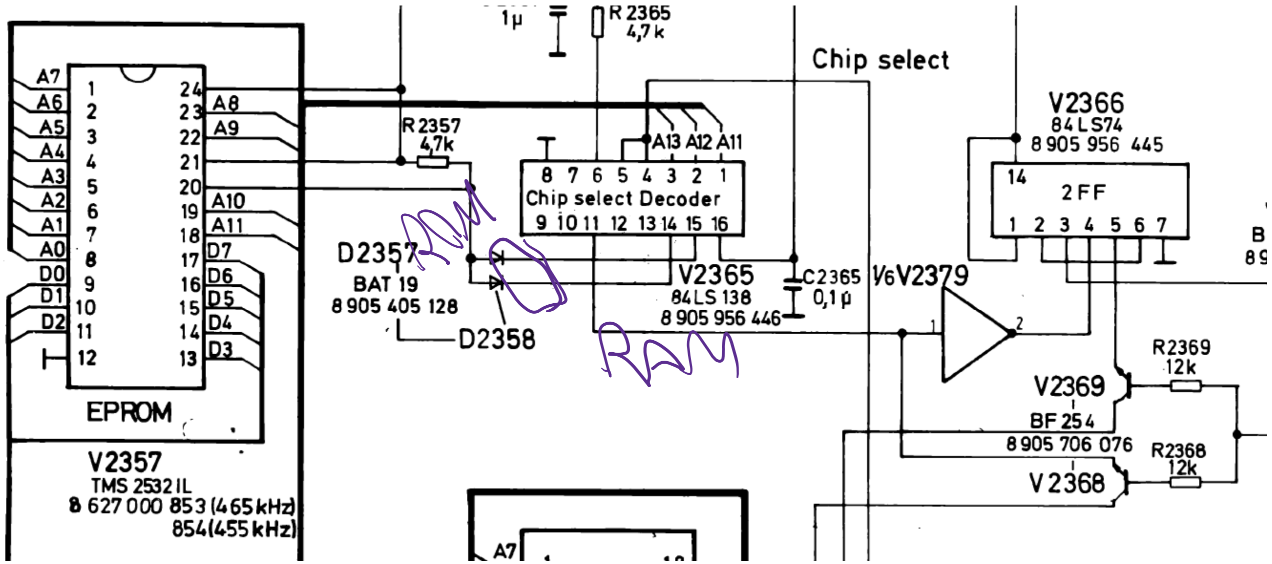
The TMS9900 series processors are big-endian. MSB is numbered 0 and comes first. LSB is numbered 15 and comes last. In the Blaupunkt schematic, however, D0/A0 are LSB and D7/A13 are MSB, even though that doesn't match the datasheet of the TMS9981.

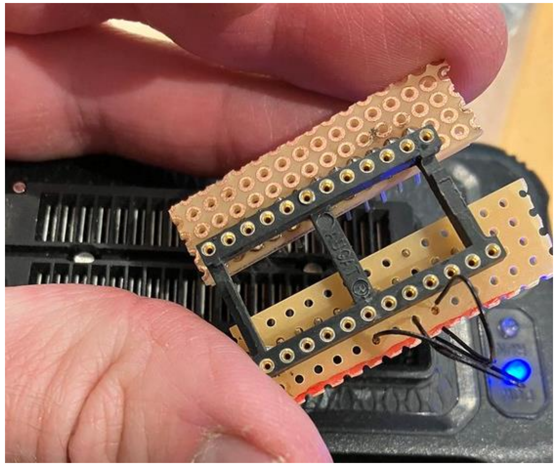
Feature request for Xgecu programmer to support TMS2532:
http://forums.xgecu.com/archiver/?tid-58.html

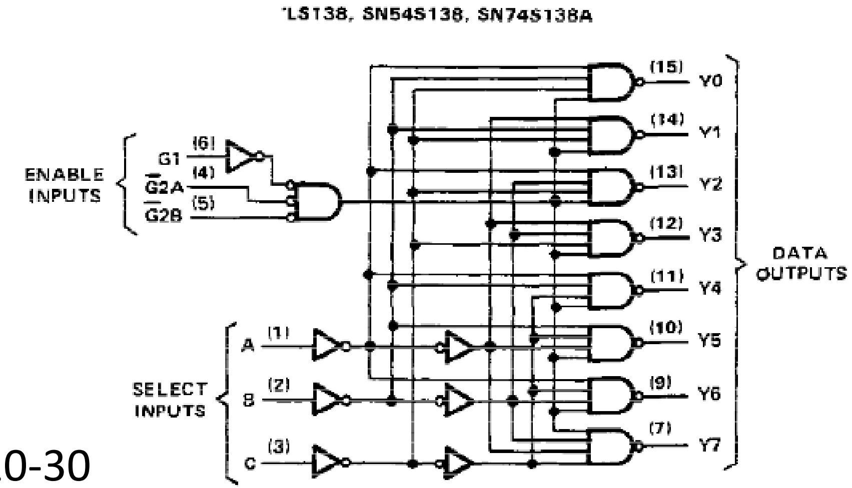
2023-10-30

Reading out the code

I made an adapter and read out the code… how is the memory map?
Chip select decoder of the circuit works like this
A13 = 0 and A12 = 0 => enable ROM, meaning that the 4K ROM is mapped from 0000-0FFF
A13 = 1 and A12 = A11 = 0 => enable RAM, meaning 256 byte RAM is mapped from 2000-3FFF





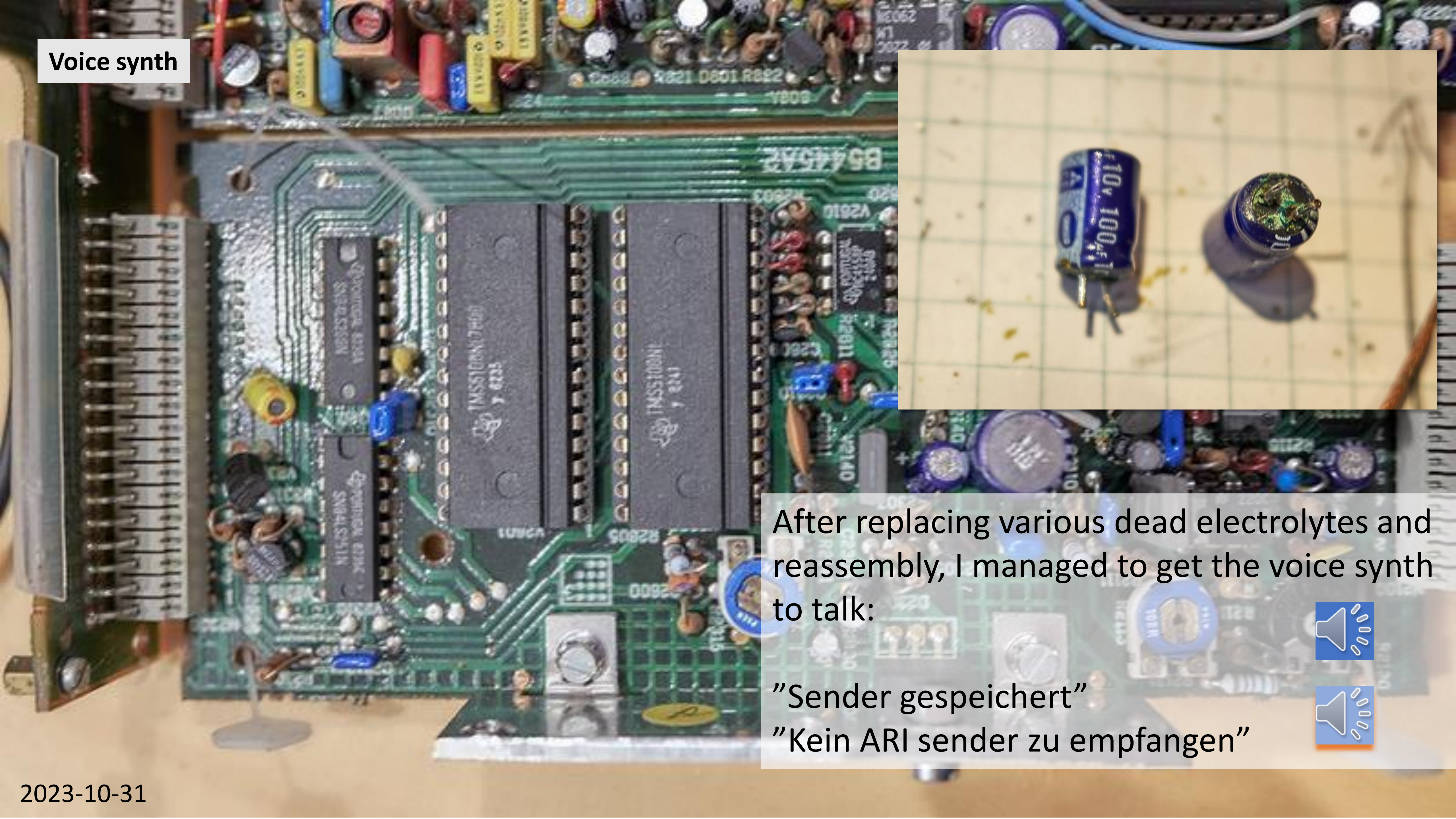


Interrupt vectors:
00000000:
2000 => RESET WP
088a => RESET PC
20e0 => INTERRUPT 1 WP
041e => INTERRUPT 1 PC
203e => INTERRUPT 2 WP
0236 => INTERRUPT 2 PC
203e => INTERRUPT 3 WP
022c => INTERRUPT 3 PC
0016 => INTERRUPT 4 WP
2074 => INTERRUPT 4 PC

This is odd?

2023-10-30

Voice synth



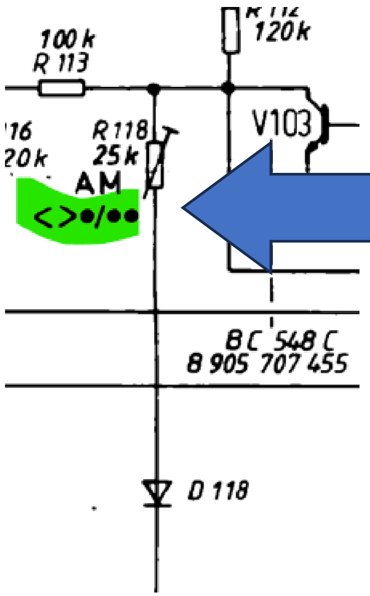
After replacing various dead electrolytes and reassembly, I managed to get the voice synth to talk:

”Sender gespeichert”
”Kein ARI sender zu empfangen”

2023-10-31

Why is the radio not working?

- At the core of the circuit is the TDA 1072 AM Receiver.
- AM PLL seems to work as frequency at PIN 10 = dialed frequency + 450 KHz exactly
- However there is no audible signal at the audio ouptut pin 6, only noise.
- A longer aerial helps, but still no sign of an audio signal, only noise.
- This is recorded directly from PIN 6



This symbol is used in the diagram, but I'm unsure what it means. It's controlled by the computer. It could be a sensitivity setting that I should play with? The trimmer shown here is in the input stage before the TDA 1072.

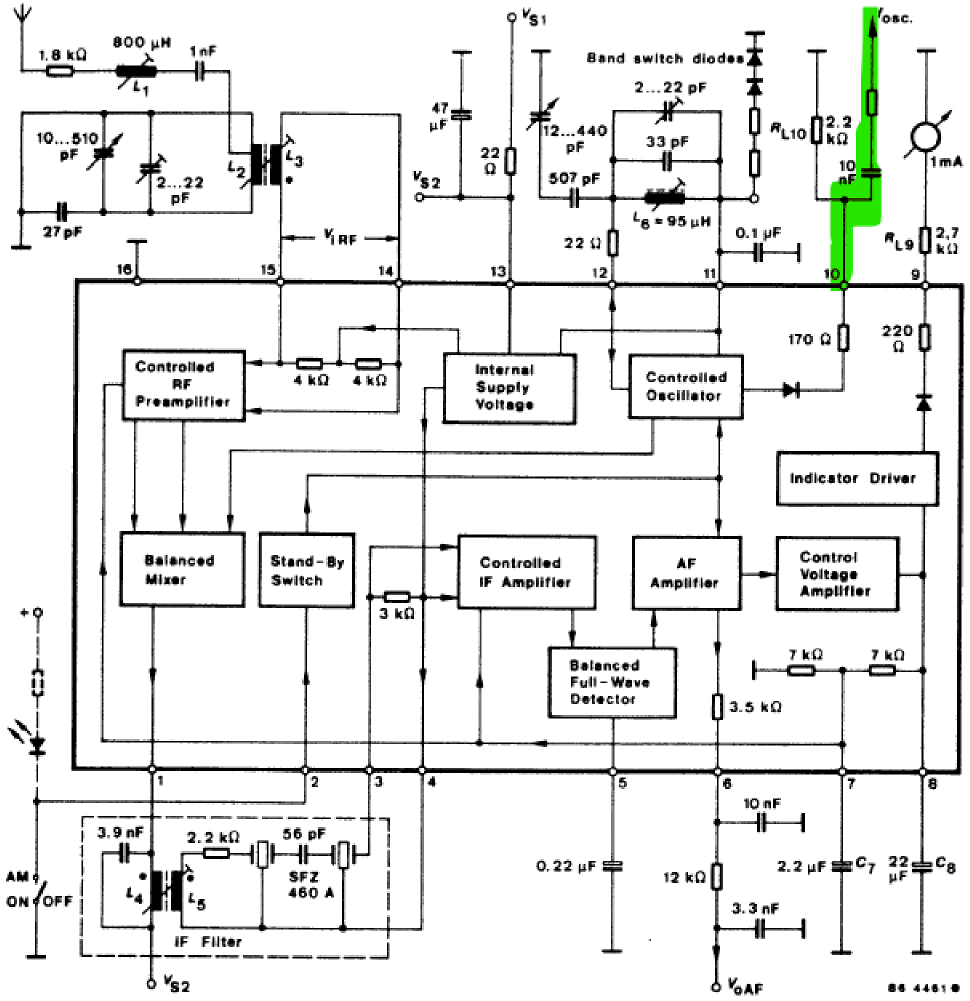


Figure 1 Block diagram and application circuit

Instruction manual says:

Empfiglichkeitsshalter:

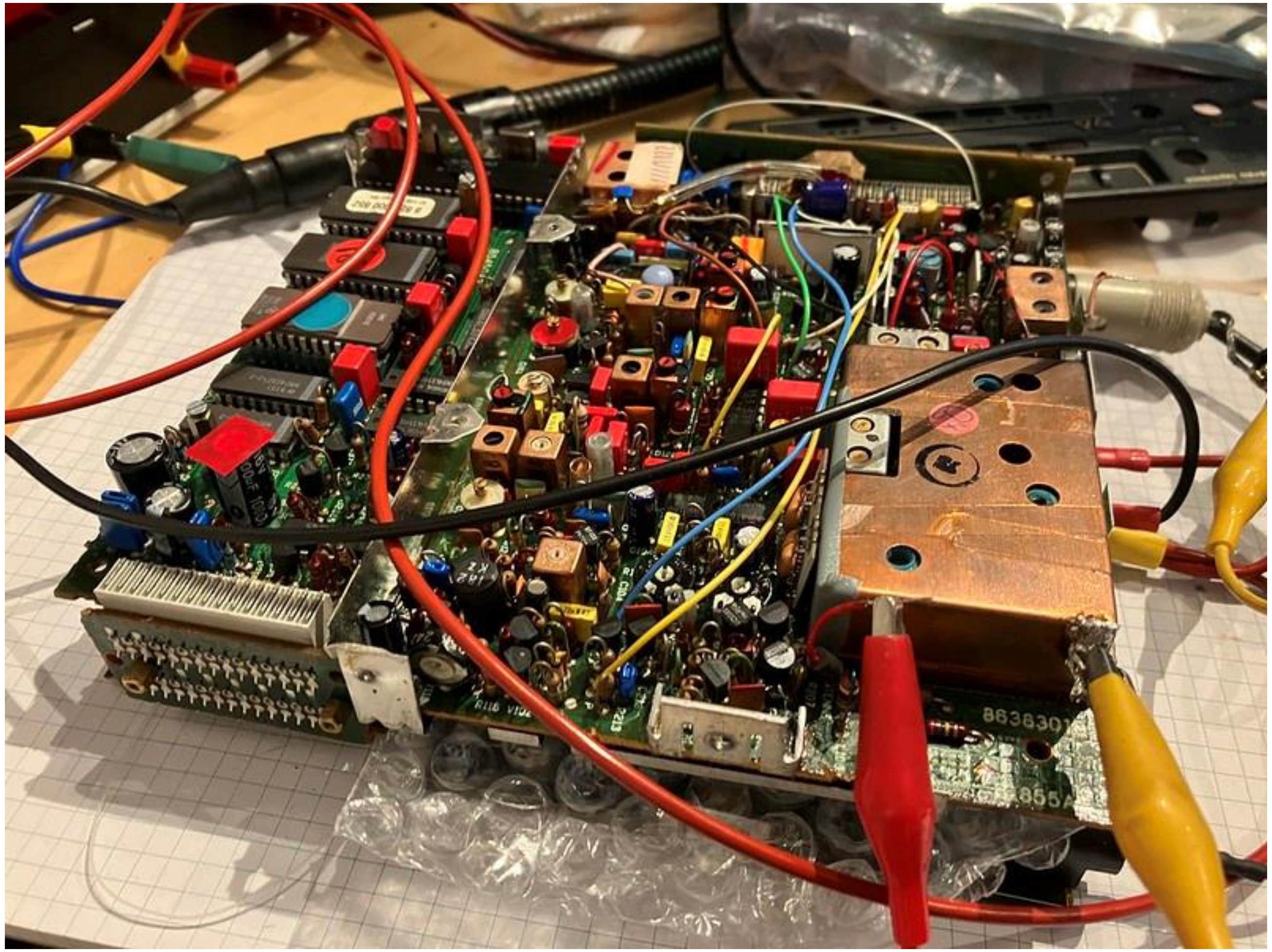
* Sie empfangen per Suchlauf nur starke Sender (normalempfindlich)

** Sie empfangen per Suchlauf all im Empfangsebiet möglichen Sender (hochempfindlich)

2023-11-01

The AM radio is working weakly!

I got a very weak reception on 225 kHz, Polish 1000 kW transmitter!
I have ordered an original repair and calibration manual for the radio on eBay.



2023-11-01

Control panel serial interface analysis

2023-11-04 edited 2023-11-07

There is apparently no clock on the data lines so the communication will be running at some fixed baud rate. The serial I/O of the COP4 cpu seems to be used and is based on it generating some interrupt. When powered on, the interrupt signal is triggered at a frequency of between 8 kHz in average. The following two paragraphs are from the COP404 data sheet:

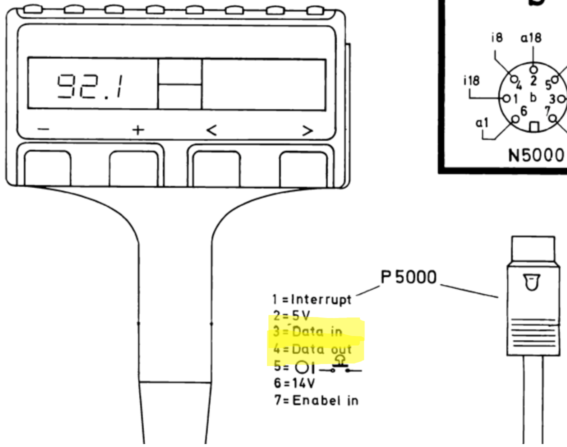
XAS INSTRUCTIONS

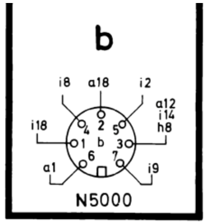
XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." The table below provides a summary of the modes associated with EN_3 and EN_0.

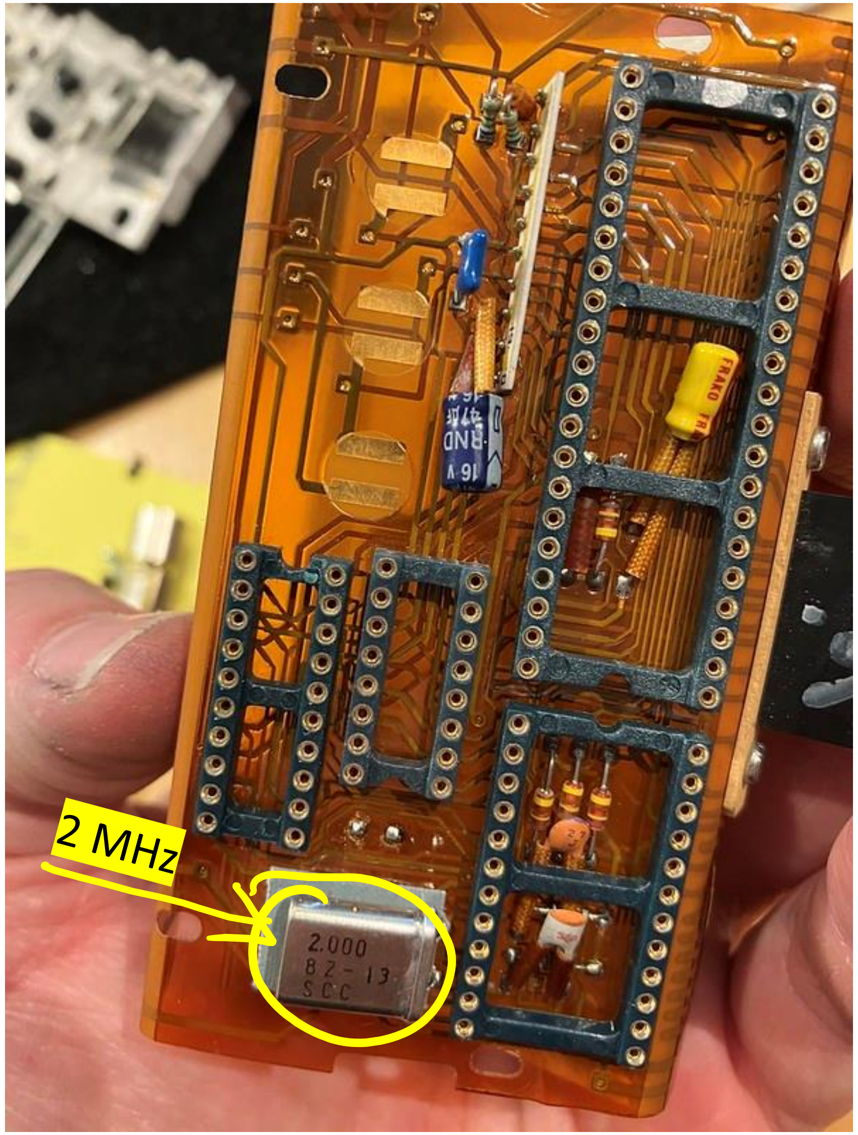
From this it can be concluded that the serial I/O is asynchronous with a bit rate derived from COP4 instruction cycle frequency. The instruction cycle time equals the crystal frequncy divided by 32 (COP404 datasheet) so it should run at at bit rate of 62.5 kHz. The interrupt signals the bit clock.

Looking at the disassembled COP4 code, this seems plausible. XAS instructions are used in several places to trigger output/input of 4 bit frames. When repeated, there are 3 instructions between. How is data in to the panel handled?






```
        OGI     10      ; output decimal 10 to G port
        LEI     a       ; enable EN3, disable EN2, EN1, EN0 this means enable SO output to output SIO serial shift register
                        ; interrupt is enabled, L drivers are disabled L port is tri-state.
L76:    LD      1
        XAS
        LD      1
        NOP
        NOP
        XAS
        XDS     0
        JP      $L76    ; 0x28E
        RC
        XAS
        OGI     8
        LEI     1
        JMP     $L1     ; 0x00A
```



Researching an alternative radio module

There are two available DAB+/FM Arduino compatible radio modules:

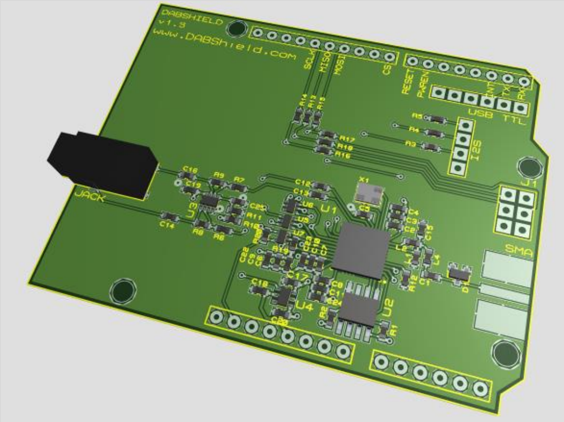
- DABShield
- Keystone T4a tuner interface

Idea is…

- Arduino Due to replace the TMS9918 based central computer.
- DAB+/FM module replaces existing LMKU radio
- Existing 4 bit bidirectional asynchronous serial interface to control panel via Arduino
- Existing TMS9918 CRU-based interface to speech synthesizer via Arduino.
- Existing data interface to the tape deck. This is unidirectional and probably only carries info on volume and pause to mute for ARI-messages

DABShield:
www.dabshield.com

- UK based, own developed shield.
- Looks well supported with good guidance and forum
- Reasonably priced at 50 GBP
- Fits directly on top of an Arduino
- Sensitivity is unknown?
- Some complain of the quality of the output signal, but this seem to be down to loading of the output due to small output capacitors which will not be a problem in my application.



Keystone/Excitron
https://excitron.be/en/products-2/dab-project/

- A belgian project which also looks serious.
- The T4A tuner is a commercial product. Excitron has developed the Arduino interface board.
- Sensitivity is good and the board supports 3.3V Arduino interfaces, like the Arduino DUE (ARM based).

2023-11-09
2023-11-18

Bluetooth module LN-BT02

The LN-BT02 is ideal for integration in a car radio since it has an onboard LM317 voltage regulator, an AUX input, a mono microphone input, and a button board with volume and play/pause buttons.

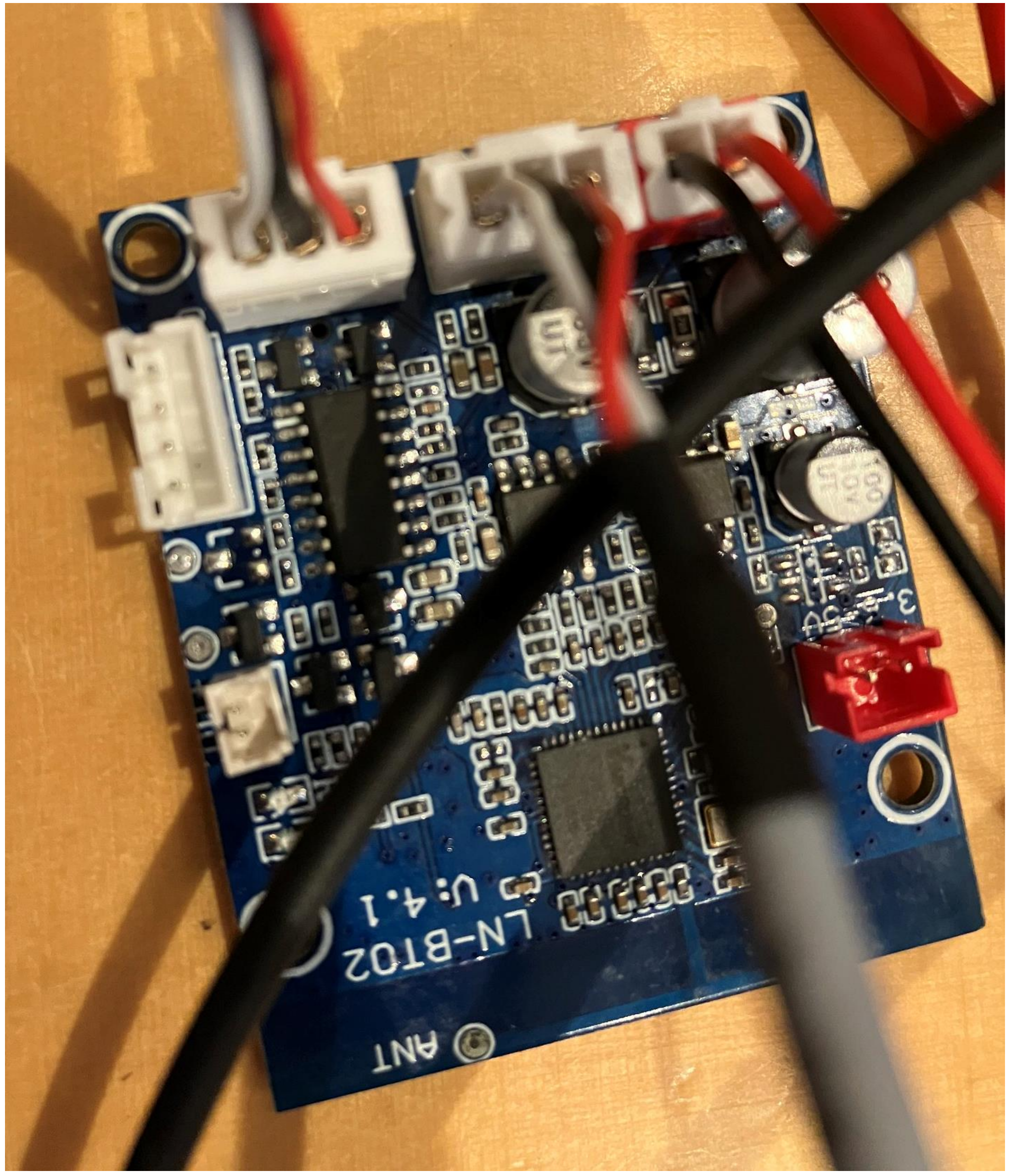
The documentation is very sparse, but I have tested it and determined that the AUX input is activated when blootooth music is paused so the unit can be fitted in-line between the tuner and tape, or inside the tape unit.

It has a MUTE output which I can see becomes enabled when the bluetooth is active. This is probably not useful. For traffic info, it is necessary to be able to stop playing and switch to the AUX input and that is toggled when activating the middle button S2.

There are two LED’s:

- RED - ???
- BLUE – blinks slowly when bluetooth signal is active, rapidly when et can be connected to, and lights up when bluetooth is paused.

Update: I have installed the module in my Blaupunkt Paris and it works perfectly. The AUX input is fed with a DC offset, but this does not cause any issues. The output drives the tone control through 1uF caps. There is no noise or clicking. The module emits a gentle beep when it connects but doesn’t interrupt radio/tape until music starts playing over bluetooth. The volume level is ok. The microphone is not connected to the module and call over bluetooth must be disabled on the phone.



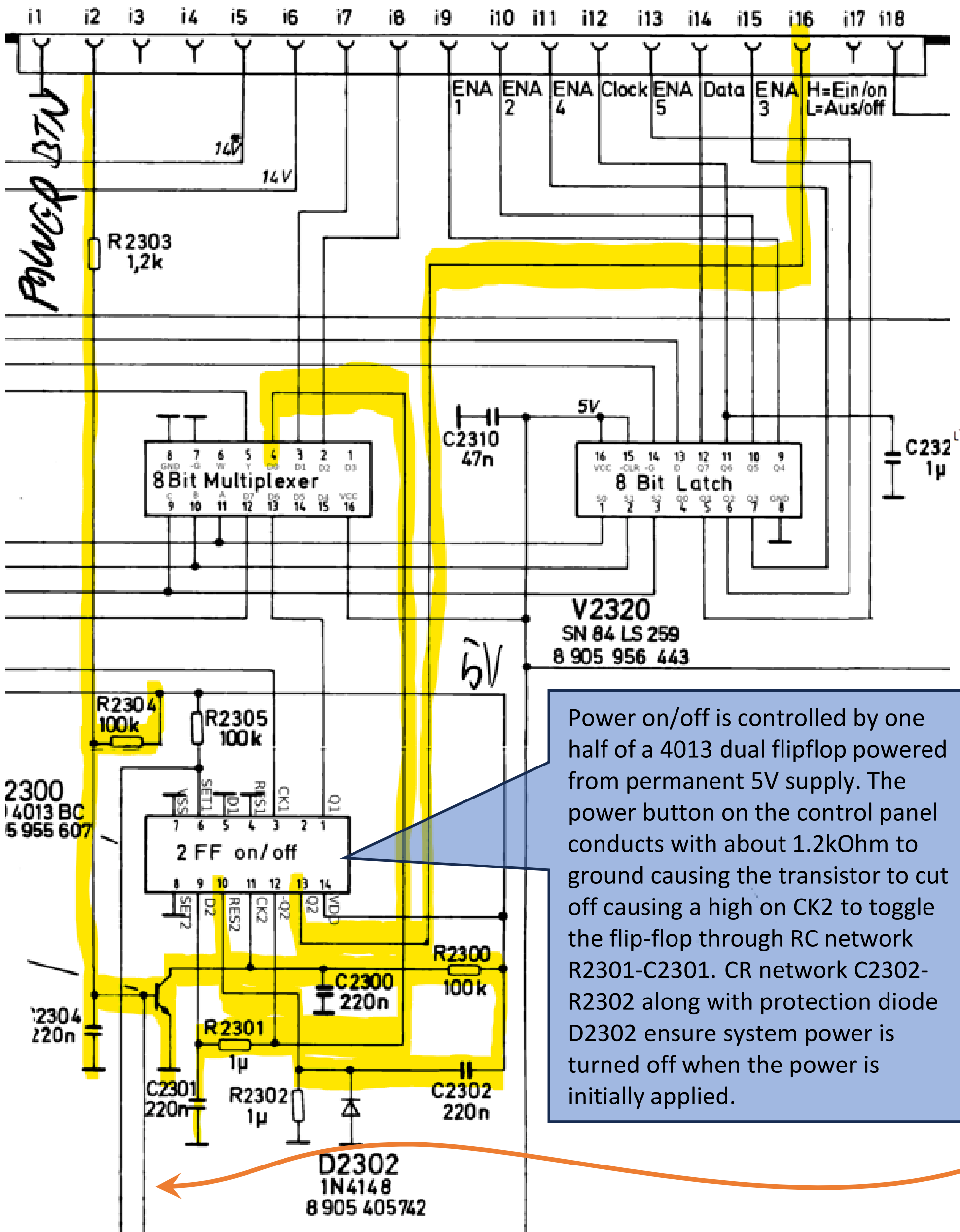
2023-11-12

Power on/off control

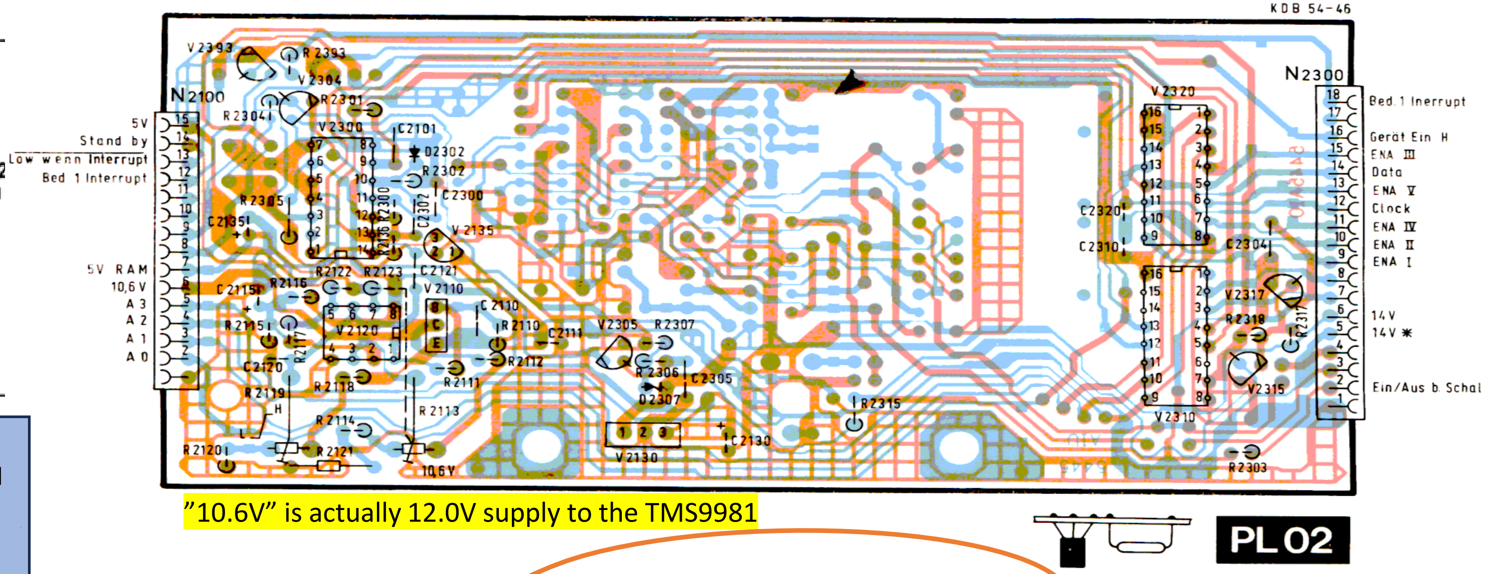
While checking power supply voltages to the radio, I ran into a problem with power switching off the unit every time I touched the 8V supply to the AM radio with my multimeter. Also, I could not switch on when I had the probe on. This lead me to take a closer look at the power on/off control circuit.

Possible root causes:

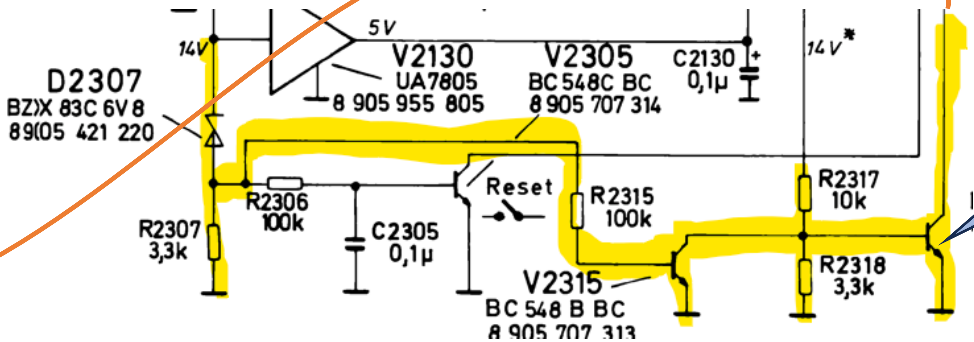
- Mechanical issue with the power button in the control panel
- Electrical problem around zener diode D2307
- Electrical problems around the CR netowrk C2302-R2302 causing spurious power off
- Problem with the 8V supply?



Power on/off is controlled by one half of a 4013 dual flipflop powered from permanent 5V supply. The power button on the control panel conducts with about 1.2kOhm to ground causing the transistor to cut off causing a high on CK2 to toggle the flip-flop through RC network R2301-C2301. CR network C2302-R2302 along with protection diode D2302 ensure system power is turned off when the power is initially applied.



”10.6V” is actually 12.0V supply to the TMS9981



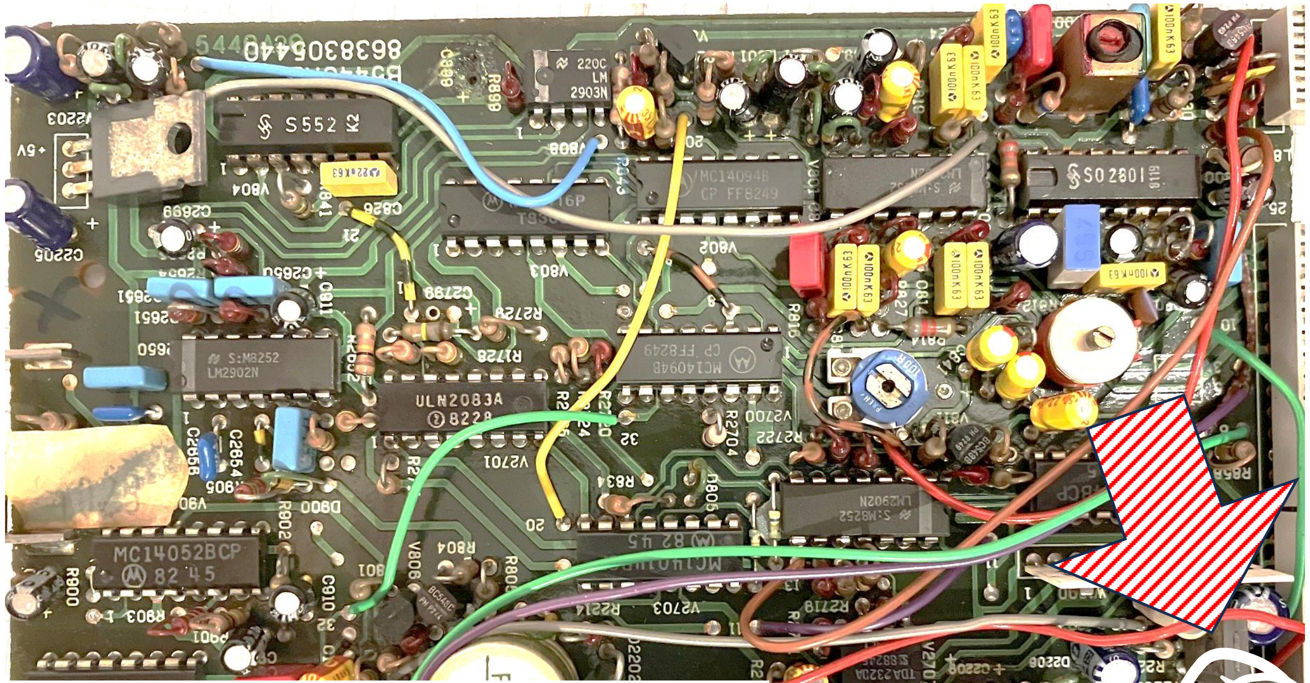
This transistor toggles system power when the 14V battery supply drops below 0.6V+6.8V=7.4V

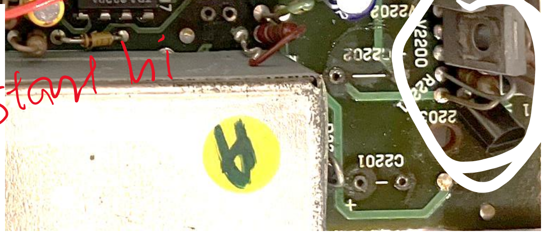
Power-on problem

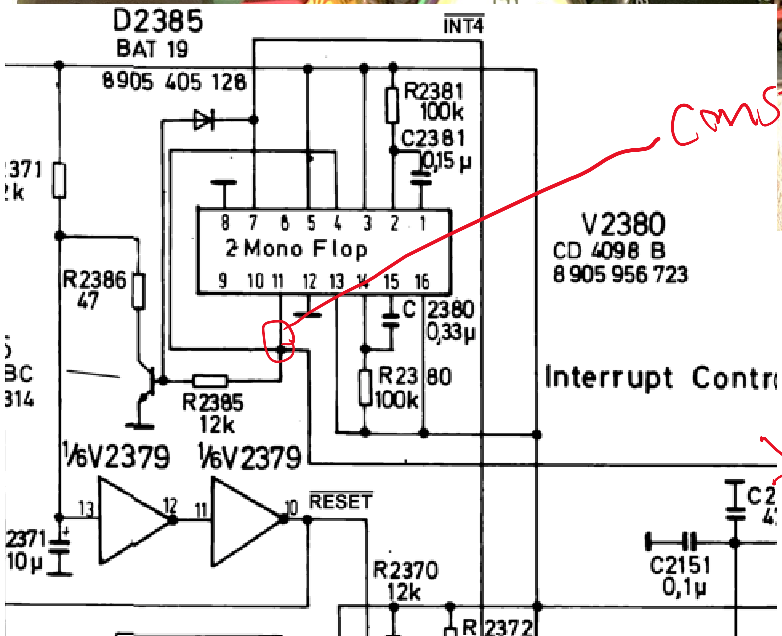
2023-11-16

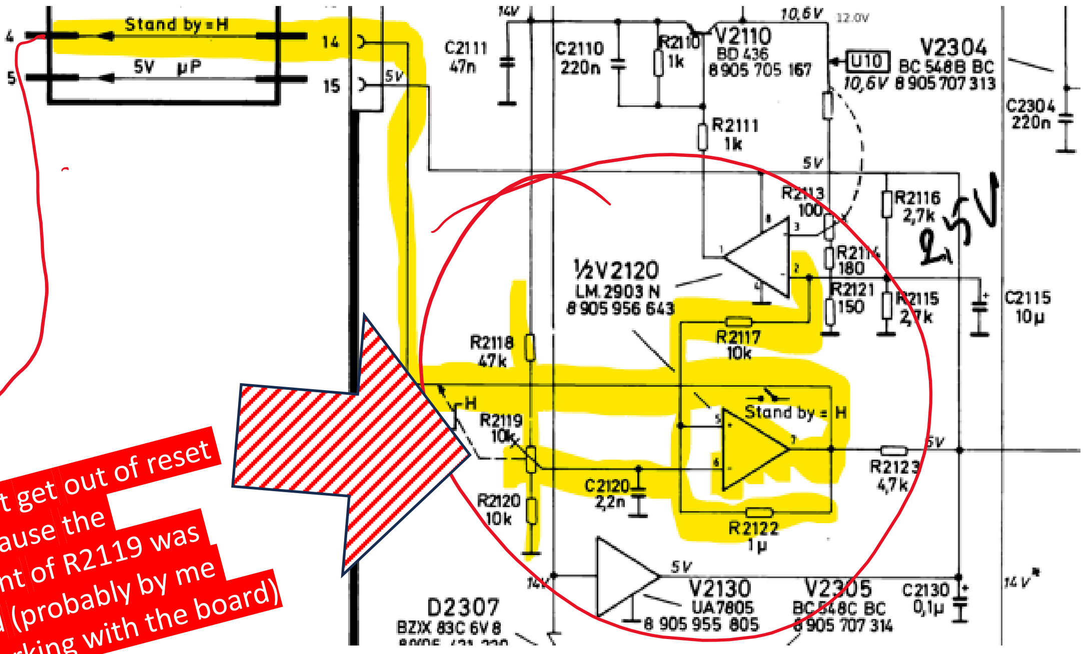
The problem with the 8V supply seems to be mechanical. However, inspecting the main connection board with the power suply, there are several cooked electrylites that have leaked their stuff on the board. They need to be replaced. A new problem has developed and the system does not power on now! What's wrong must be investigated.

It turned out the problem was the V2201 transistor and resistors R2201 and R2200. The transistor was bent over because of the DIN connector to the control panel. Also the resistors were touching each other causing a short circuit. Once this was rectified the system could power on- and off and remain on even when touching parts.



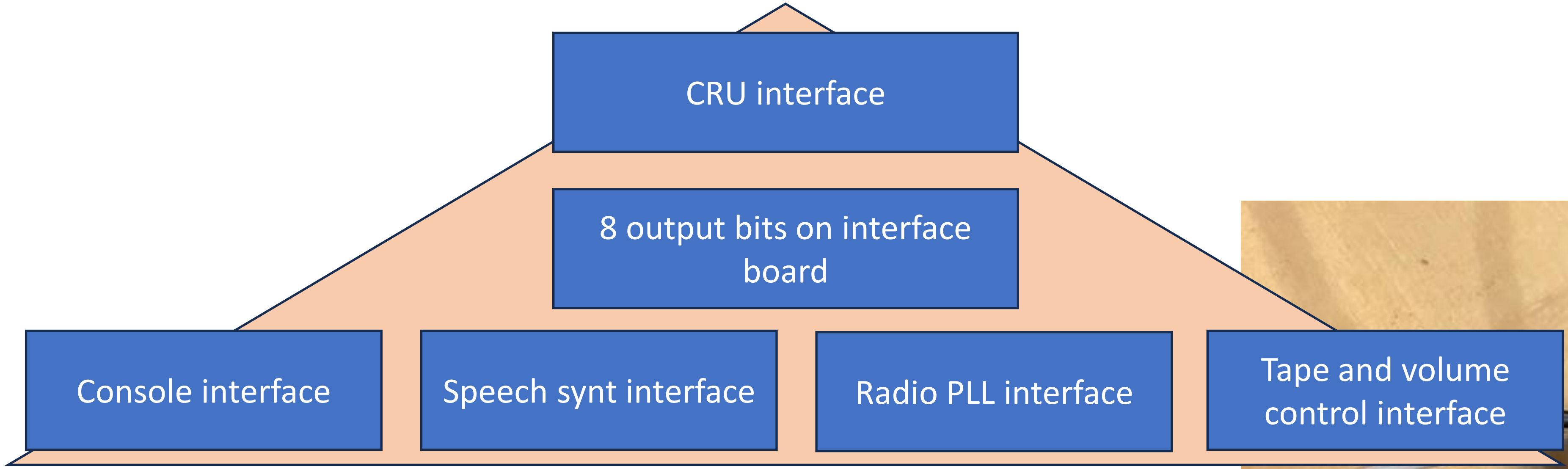






CPU did not get out of reset status because the adjustment of R2119 was distorted (probably by me while working with the board)

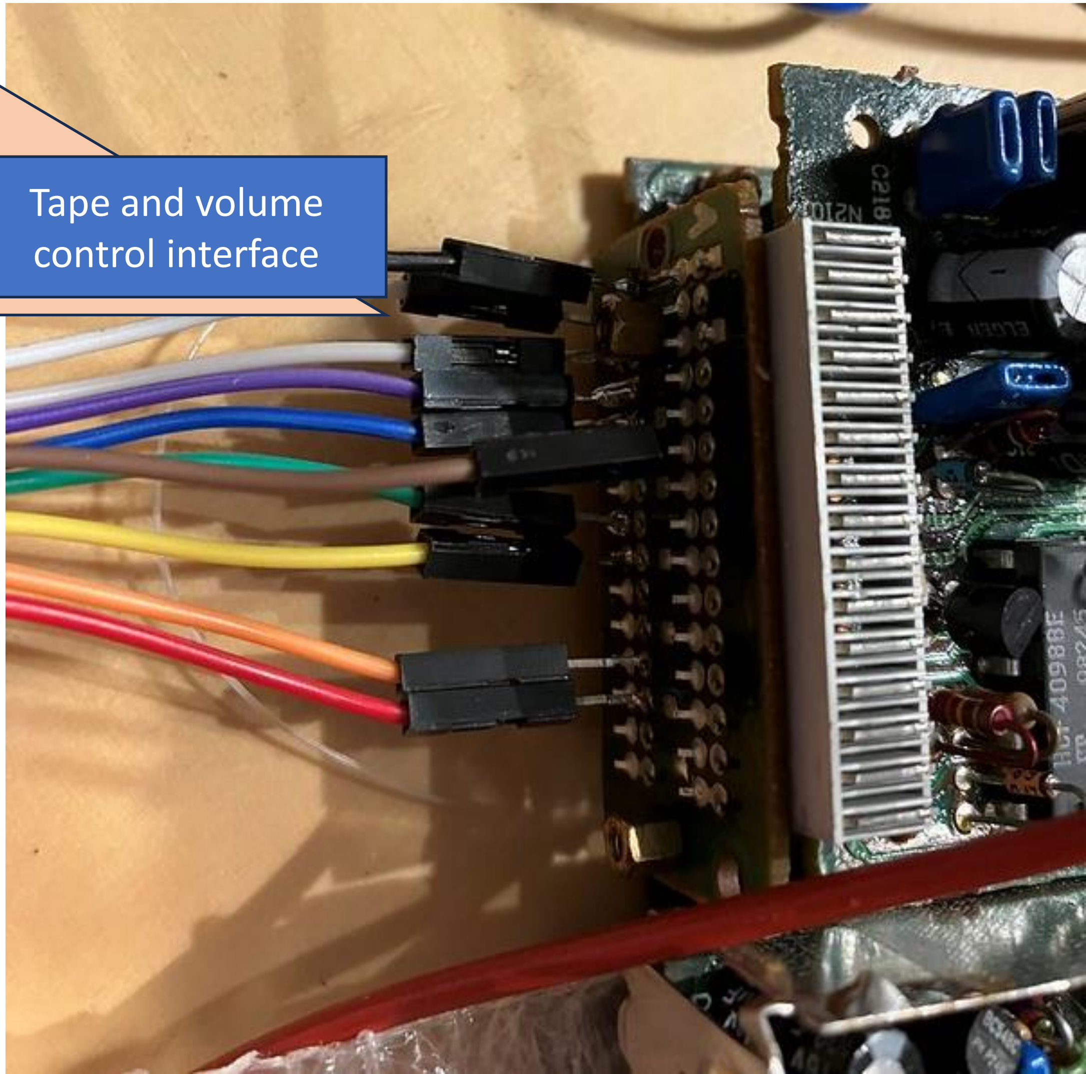
Reverse engineering protocols



I am tapping into and recording the I/O from the CPU board with an USB logic analyzer recording at 4 MHz.

This is essentially CRU interface and the data takes some work to interpret since the data is ”noisy” with address bus activity going on at the same time. Also detecting the timing of input data needs to be heuristic as there is no clock for inputting data. The CPU just reads the CRU input whenever it wants to without signalling that to the outside world.

Also the interfaces are layered as shown in the model on top so interpreting the data needs to consider that.



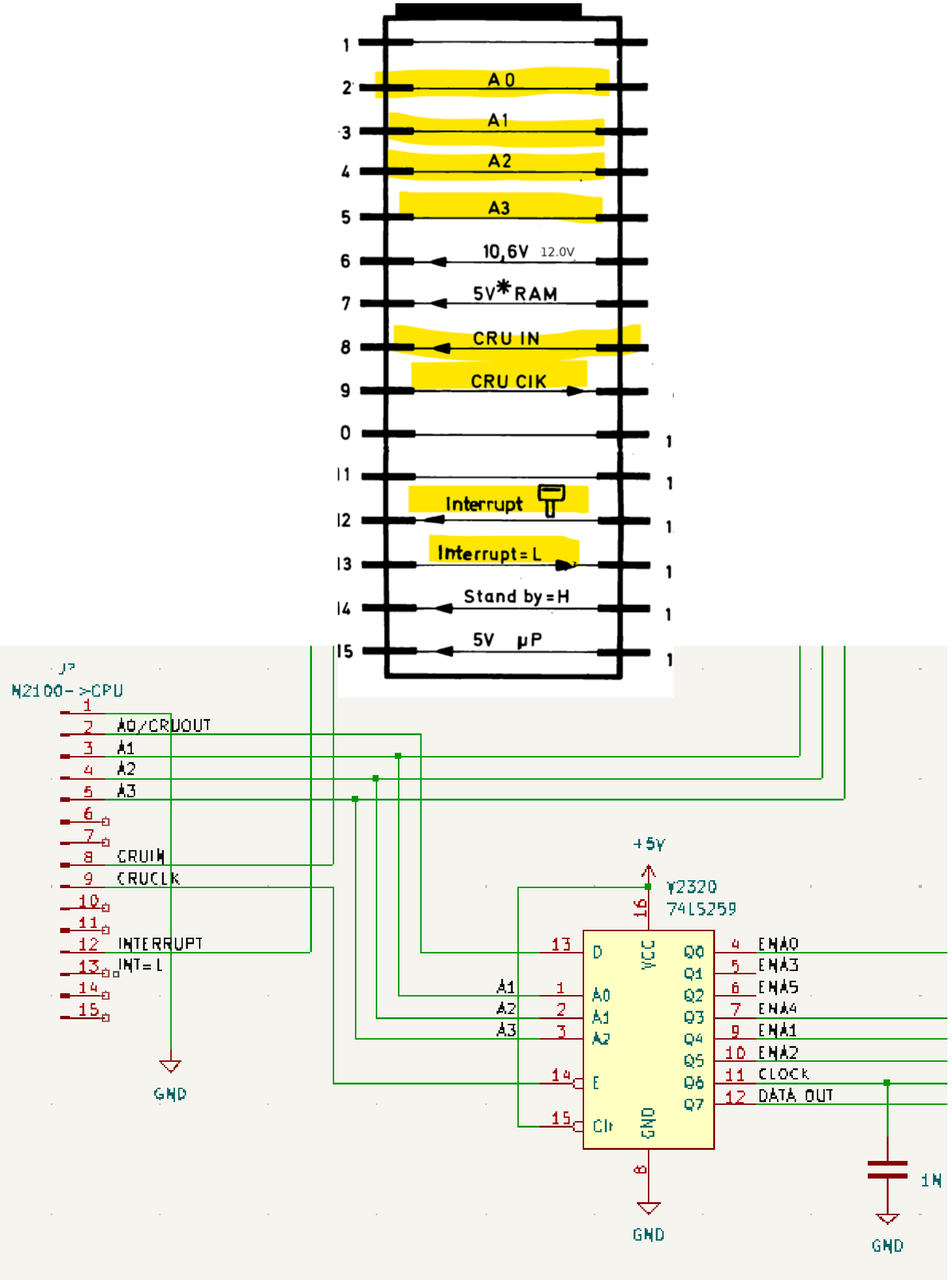
- 2023-11-21

Reverse engineering interfaces

I am tapping into and recording the eight yellow marked signals. My plan is to write a C program that interprets the data.

The table below outlines the CRU I/O connections to interfaces. All interfaces are via shift registers. The shcematic on the lower right shows the 74LS259-based CRU out decoder and is redrawn from the schematics.

I know that the interrupt from the operator panel is just a single interrupt per keypress.



CRU PORT	IN	OUT
0	~POWER ON	ENA0 = Enable SR IF TM5100 speech synthesizer
1	DATA IN	ENA3 = ARI decoder
2	Operator panel DATA IN	ENA5 = PLL
3	N/C	ENA4 = Tape and various status inputs
4	N/C	ENA1 = Operator panel
5	N/C	ENA2 = Output MUX + cassette/volume
6	POWER RESET	DATA CLOCK
7	INTERRUPT=L	DATA OUT

2023-11-21

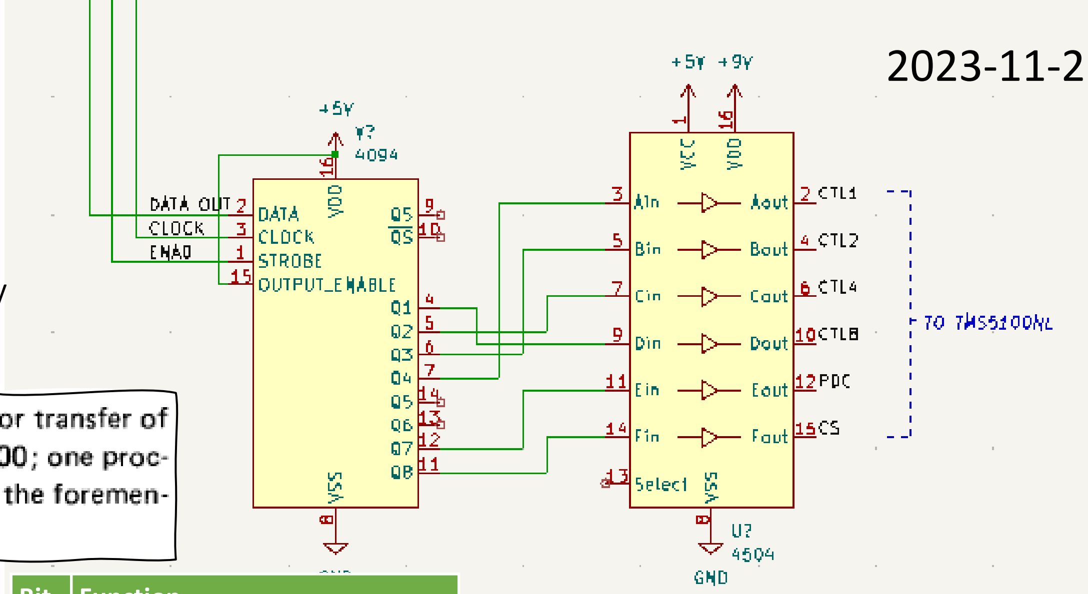
Speech synthesizer interface

2023-11-26

The TMS 5100 interface consists of a 4094 shift register connected to the CRU decoder through three lines: The dedicated ENA0 and the system-shared DATA OUT and CLOCK lines. A 4504 level converter converts the TTL levels to the 9V P-MOS levels needed by the TMS 5100.

The schematics on the right shows the interface redrawn by me. The code below shows the disassembly of the CPU procedure used to output the 4 bits of command plus PDC and CS signals to the speech synthesizer. Signals are explained like this in the TMS 5100 info I have.

> The TMS 5100 has a six line control interface partitioned as follows: four bidirectional lines CTL 1-8 for transfer of commands and ROM addresses to the TMS 5100, or of speech status, or of ROM data to the TMS 5100; one processor data clock line (PDC) to transfer the data on CTL 1-8; and, one chip select (CS) line, to enable the foremen-tioned five lines.




```
XOP4  LIMI    >0001           ; pc:>00ee w:>0300  Send a command to the speech synthesizer interface
      MOVB    *r11+,r5        ; pc:>00f2 w:>d17b  R11 (XOP parameter) points the command to send
      MOVB    *r11+,r4        ; pc:>00f4 w:>d13b
      SRL     r4,8            ; pc:>00f6 w:>0984
      MOV     *r11+,r1        ; pc:>00f8 w:>c07b
      MOV     *r11,r12        ; pc:>00fa w:>c31b
      SBZ     0               ; pc:>00fc w:>1e00  ENA0 = 0 => get ready to strobe data out
      MOV     r11,r9          ; pc:>00fe w:>c24b
      LI      r12,>0000       ; pc:>0100 w:>020c
      LI      r2,>0001        ; pc:>0104 w:>0202
      MOV     *r1+,r3         ; pc:>0108 w:>c0f1
      COC     r2,r3           ; pc:>010a w:>20c2  Shift out a bit
      JEQ     >0112           ; pc:>010c w:>1302  Bit is a ZERO
      SBZ     7               ; pc:>010e w:>1e07
      JMP     >0114           ; pc:>0110 w:>1001
      SBO     7               ; pc:>0112 w:>1d07  Bit is a ONE
      SBO     6               ; pc:>0114 w:>1d06  Trigger the clock = 1
      SBZ     6               ; pc:>0116 w:>1e06  Clock back to 0
      DEC     r4              ; pc:>0118 w:>0604
      JEQ     >0122           ; pc:>011a w:>1303  Are we done?
      SLA     r2,1            ; pc:>011c w:>0a12
      JEQ     >0104           ; pc:>011e w:>13f2
      JMP     >010a           ; pc:>0120 w:>10f4
      SBZ     7               ; pc:>0122 w:>1e07  We're done…
      MOV     *r9,r12         ; pc:>0124 w:>c319
      SBO     0               ; pc:>0126 w:>1d00  ENA0 = 1 => strobe data out
      MOVB    r5,r5           ; pc:>0128 w:>d145
      JEQ     >012e           ; pc:>012a w:>1301
      SBZ     0               ; pc:>012c w:>1e00
      RTWP                    ; pc:>012e w:>0380  Return
```

Bit	Function
0	CTL1
1	CTL2
2	CTL4
3	CTL8
4,5	Unused
6	PDC = processor data clock
7	CS = chip select

Documentation of the TMS5100 command set has been lost but a reverse engineering effort of the Speak-and-Spell play-tool (which it was used in) has been done by furrtek. From this, the below has been derived, so I now know what to look for. I need to find the addresses of words and sentences in the TMS6100 ROM.

Dec	CTL8	CTL4	CTL2	CTL0	Description
10	1	0	1	0	Start talking. Play the bitstream from the ROM.
0	0	0	0	0	NOP. No operation, do nothing.
14	1	1	1	0	Read status. At the next second pulse on PDC, bit 0 (CTL1) becomes an output and indicates whether the synthesizer is talking or not.
2	0	0	1	0	Load 4 address bits in the ROMs.
8	1	0	0	0	Read a bit from the ROM.
4	0	1	0	0	Read 4 bits register with data from the ROM.

To set the address, we have to make 5 writes, to set 18 bits + 2 which are ignored. After each write, an internal pointer advances 4 bits to point to the next ones in the address register. This pointer is reset to zero only after a bit read.

Write #	Address bits set
1	A3~A0
2	A7~A4
3	A11~A8
4	CS1,CS0,A13,A12
5	ignored,ignored,CS3,CS2

Source: http://furrtek.free.fr/index.php?a=speakandspell&ss=4&i=2

Analyzing actual speech synthesizer commands

2023-12-03

I have analyzed data from the logic analyzer using a custom C-program that reads the raw data dump from the logic analyzer, extracts CRU-OUT data and generates a listing of events and data. This has allowed me to extract a series of data being sent to the TMS5100.

```
1729081us   114us                          ENA4                 DOUT=1
1729108us    27us                          ENA4         DCLOCK  DOUT=1
1729124us    16us                          ENA4                 DOUT=1
1729238us   114us                          ENA4                 DOUT=0
1729266us    28us                          ENA4         DCLOCK  DOUT=0
1729282us    16us                          ENA4                 DOUT=0
1729396us   114us                          ENA4                 DOUT=0
1729424us    28us                          ENA4         DCLOCK  DOUT=0
1729440us    16us                          ENA4                 DOUT=0
1729554us   114us                          ENA4                 DOUT=0
1729582us    28us                          ENA4         DCLOCK  DOUT=0
1729598us    16us                          ENA4                 DOUT=0
1729714us   116us                          ENA4                 DOUT=1
1729730us    16us                          ENA4         DCLOCK  DOUT=1
1729746us    16us                          ENA4                 DOUT=1
1729860us   114us                          ENA4                 DOUT=0
1729888us    28us                          ENA4         DCLOCK  DOUT=0
1729904us    16us                          ENA4                 DOUT=0
1730018us   114us                          ENA4                 DOUT=0
1730046us    28us                          ENA4         DCLOCK  DOUT=0
1730062us    16us                          ENA4                 DOUT=0
1730106us    44us                          ENA4                 DOUT=0
1730150us    44us      E0=c4               ENA4                 DOUT=0  TMS5100(CS=1 PDC=1 CTL=2)
1730198us    48us                          ENA4                 DOUT=0
1730558us   360us                          ENA4                 DOUT=0
1730698us   140us                          ENA4                 DOUT=1
1730714us    16us                          ENA4         DCLOCK  DOUT=1
1730730us    16us                          ENA4                 DOUT=1
1730844us   114us                          ENA4                 DOUT=0
1730872us    28us                          ENA4         DCLOCK  DOUT=0
1730888us    16us                          ENA4                 DOUT=0
1731002us   113us                          ENA4                 DOUT=0
1731030us    28us                          ENA4         DCLOCK  DOUT=0
1731046us    16us                          ENA4                 DOUT=0
1731160us   114us                          ENA4                 DOUT=0
1731188us    28us                          ENA4         DCLOCK  DOUT=0
1731204us    16us                          ENA4                 DOUT=0
1731318us   114us                          ENA4                 DOUT=0
1731346us    28us                          ENA4         DCLOCK  DOUT=0
1731362us    16us                          ENA4                 DOUT=0
1731476us   114us                          ENA4                 DOUT=0
1731504us    28us                          ENA4         DCLOCK  DOUT=0
1731520us    16us                          ENA4                 DOUT=0
1731636us   116us                          ENA4                 DOUT=1
1731652us    16us                          ENA4         DCLOCK  DOUT=1
1731668us    16us                          ENA4                 DOUT=1
1731782us   114us                          ENA4                 DOUT=0
1731810us    28us                          ENA4         DCLOCK  DOUT=0
1731826us    16us                          ENA4                 DOUT=0
1731870us    44us                          ENA4                 DOUT=0
1731914us    44us      E0=82               ENA4                 DOUT=0  TMS5100(CS=1 PDC=0 CTL=4)
             48us                          ENA4                 DOUT=0
```

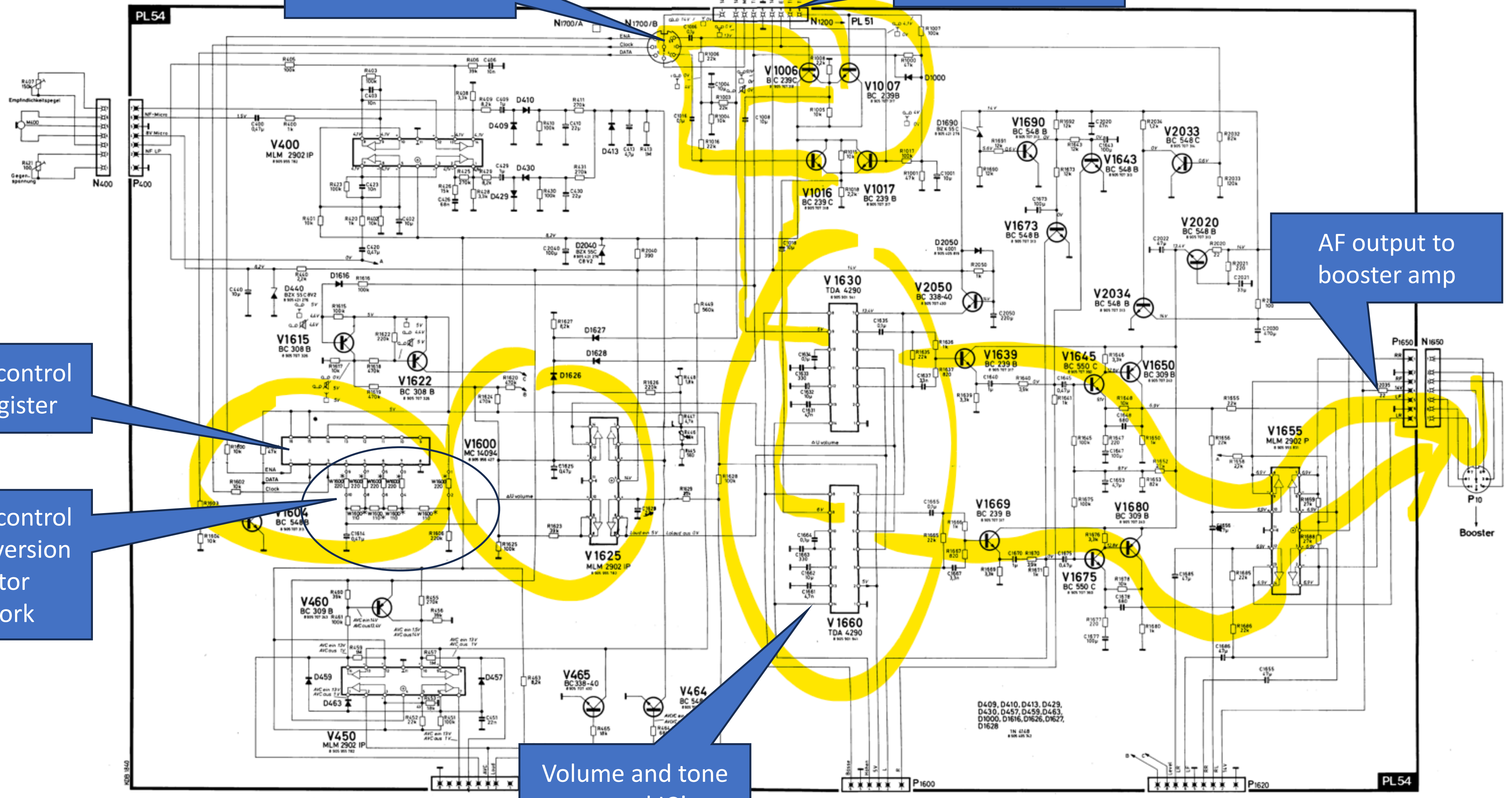
```
 868904us    E0=c4      TMS5100(CS=1 PDC=1 CTL=2) ; Load TMS6100 bit address in the next 5
 874088us    E0=c9      TMS5100(CS=1 PDC=1 CTL=9) ; A3-0             = 9
 879289us    E0=c4      TMS5100(CS=1 PDC=1 CTL=2) ; A7-4             = 2
 884492us    E0=c8      TMS5100(CS=1 PDC=1 CTL=1) ; A11-8            = 1
 889704us    E0=c0      TMS5100(CS=1 PDC=1 CTL=0) ; CS0,CS1,A13,A12 = 0
 900146us    E0=c1      TMS5100(CS=1 PDC=1 CTL=8) ; CS3,CS2,n,n      = 8
 905284us    E0=c5      TMS5100(CS=1 PDC=1 CTL=a) ; Start talking command
…
1157562us    E0=c7      TMS5100(CS=1 PDC=1 CTL=e) ; Read status?
…
1716306us    E0=c0      TMS5100(CS=1 PDC=1 CTL=0) ; No operation
```

- **This shows that my analysis of the circuit is correct,**
- The TMS5100 uses the same command set as the Speak&Spell analyzed by furrtek.
- The byte at 0129 probably says ”Sender gespeichert”.
- CS3=1, CS2=CS1=CS0=0.
- The read status commands are sent until the synthesizer has ended talking.
- I do not know why it sends a No operation later?

With this information, it should be possible to try all 16K addresses to see what it says once I have an arduino hooked up to emulate the CRU interface.

Volume control circuit elements

2023-12-03



AF input from radio

AF input from tape deck

AF output to booster amp

Volume control shift register

Volume control D-A conversion resistor network

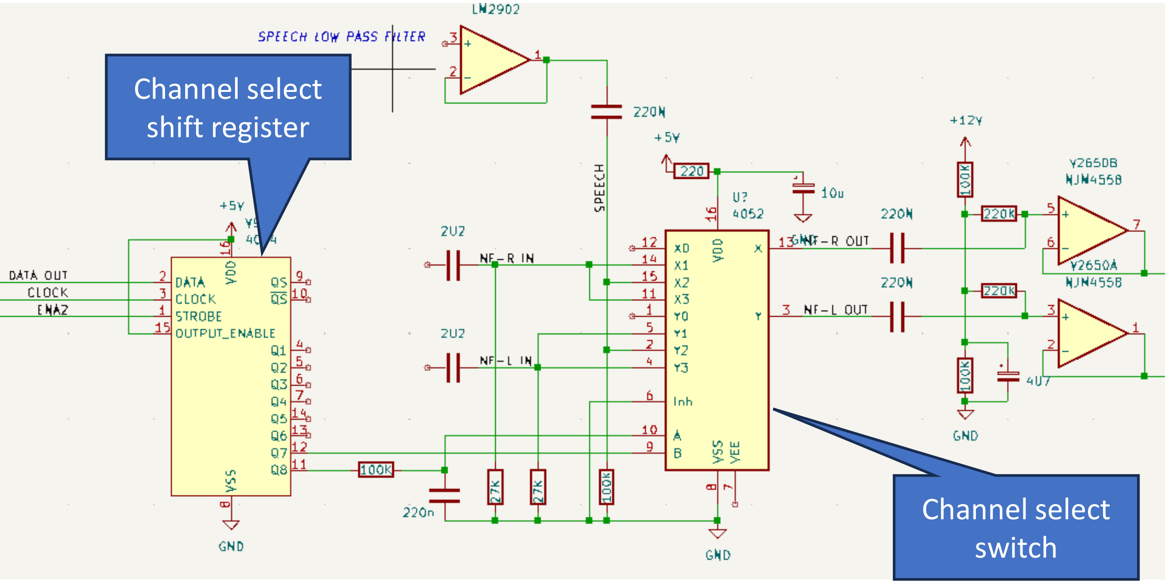
Volume and tone control IC's

Channel select and volume control analysis

2023-12-03

Channel select and volume control appears to the CPU to happen over the same shift register, but it is in fact distributed across two shift registers sharing the same data, clock, and strobe lines: Channel select happens in the head unit and volume control is embedded in the tape deck.

Channel select

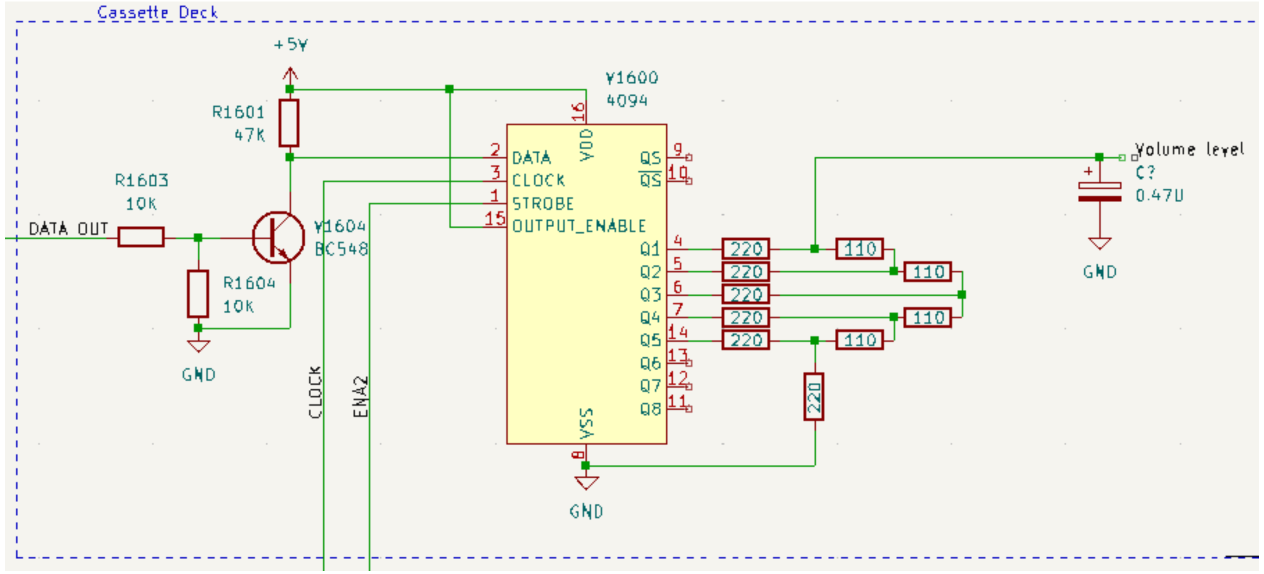


$Q8,Q7 = 00_{binary}$ => floating input
$Q8,Q7 = 01_{binary}$ => radio
$Q8,Q7 = 10_{binary}$ => speech synth
$Q8,Q7 = 11_{binary}$ => radio

Latched out with ENA2

The 220n capacitor on the Q8 output means that there is a short break in the sound when it switches.

Volume control



The ladder resistor network is explained here: https://en.wikipedia.org/wiki/Resistor_ladder
The inverter made by the input resistor reverses the output vs value. Least significant 5 bits => volume level. Volume is controlled by two TDA4290 IC's. The 0-5V output from the DAC is further processed with input from the ambient sound microphone before being fed to the 4290's. 0V output gives max attenuation in the TDA4290.

Example data captured and decoded from the logic analyzer CRU output:

```
 449257us E2=9f DOUT=0 VOL(LVL=31 CH=2)
6795583us E2=45 DOUT=0 VOL(LVL=05 CH=1)
6807604us E2=65 DOUT=0 VOL(LVL=05 CH=1)
6819624us E2=55 DOUT=0 VOL(LVL=21 CH=1)
```

Operator panel input -> CRU input oddity

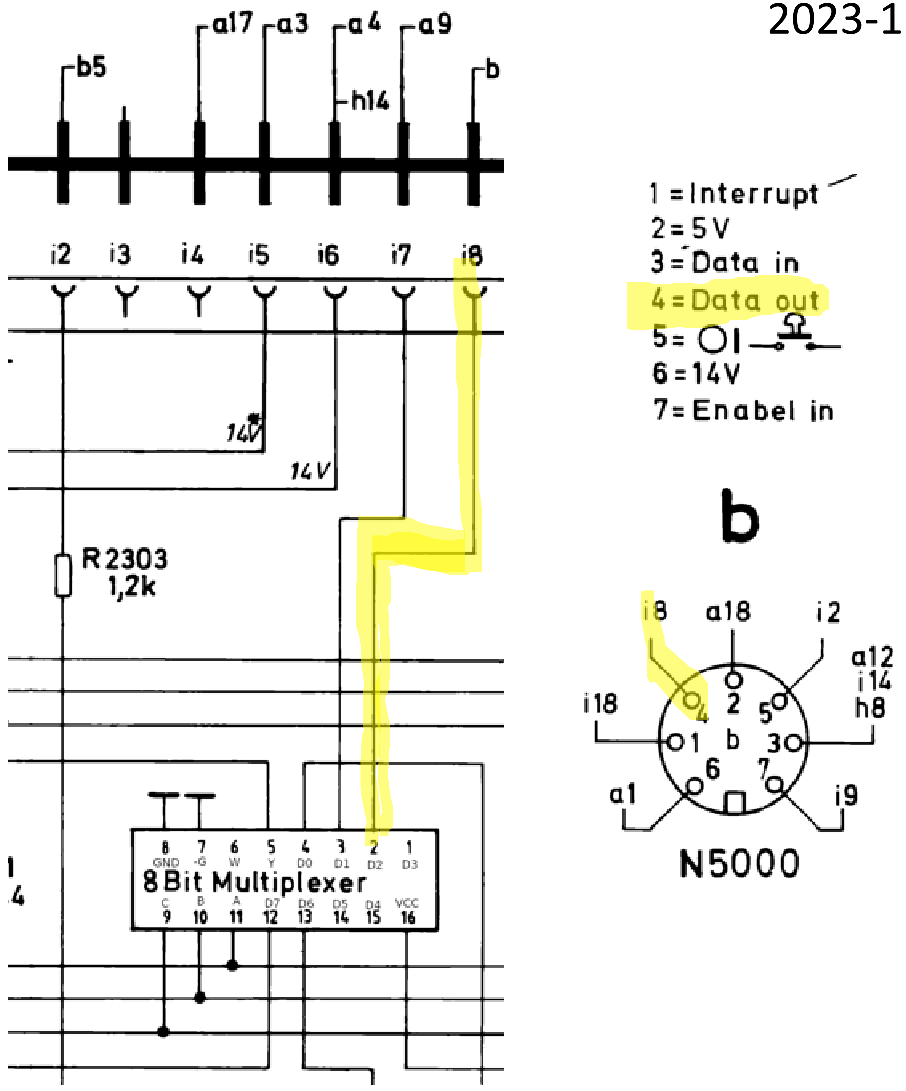
2023-11-27

There is a strange thing that according to the schematics, the operator panel input goes to CRU bit 2, however, browsing around the code, I realized there is no instruction in the code to test that bit. The Interrupt routine instead reads from CRU bit 0 which according to the diagrams are connected to the power control circuit.

So either the code doesn't work (which it does), the disassembler is wrong (it isn't), or the schematic diagram is wrong (more likely).

By the way, it looks like the code waits for a stable start bit = 1 and then reads 8 databits so it seems like a normal asynchronous serial interface with a fixed baud rate.

```
INT2        LI     r12,>0004        ; Operator panel interrupt: READ DATA
I3E         LI     r8,>03e8         ; r8 = 1000
INT2A       DEC    r8               ; do   r8 = r8 - 1
            JEQ    INT2H            ; * if r8 = 0 timeout???
            TB     0                ; * read CRU bit 0 power on???
            JEQ    INT2A            ; until CRU bit 0 = 1
            LI     r10,>0006        ; r10 = 6
            LI     r8,>0008         ; r8 = 8
INT2B       DEC    r10              ; 6 cycle delay
            JNE    INT2B            ; *
            TB     0                ; read CRU bit 0 power on???
            JEQ    INT2H            ; if CRU bit 0 <> 1 then
            LI     r10,>000b        ; * 11 cycle delay
INT2C       DEC    r10              ; * do delay *
            JNE    INT2C            ; * * *
            SRL    r7,1             ; * * shift R7 right (what is in R7???)
            TB     0                ; * * read CRU bit 0 power on????
            JEQ    INT2D            ; * * if CRU bit 0 = 0
            ANDI   r7,>7fff         ; * * * r7 = r7 AND 0111111111111111
            JMP    INT2E            ; * * else
INT2D       ORI    r7,>8000         ; * * * r7 = r7 OR  1000000000000000
INT2E       DEC    r8               ; * * r8 = r8 - 1
            JEQ    INT2F            ; * *
            LI     r10,>000a        ; * * 10 cycle delay
            JMP    INT2C            ; * until r8 = 0
INT2F       LI     r10,>000b        ; * 11 cycle delay
INT2G       DEC    r10              ; * * delay
            JNE    INT2G            ; * * delay
            TB     0                ; * read CRU bit 0 ???
            JEQ    INT2I
INT2H       CLR    r7
INT2I       LI     r12,>0000
INT2J       TB     7
```



Baud rate calculation requires that I handtrace the instructions executed and compares with the instruction execution times in TABLE 4 in the TMS9981 data sheet….

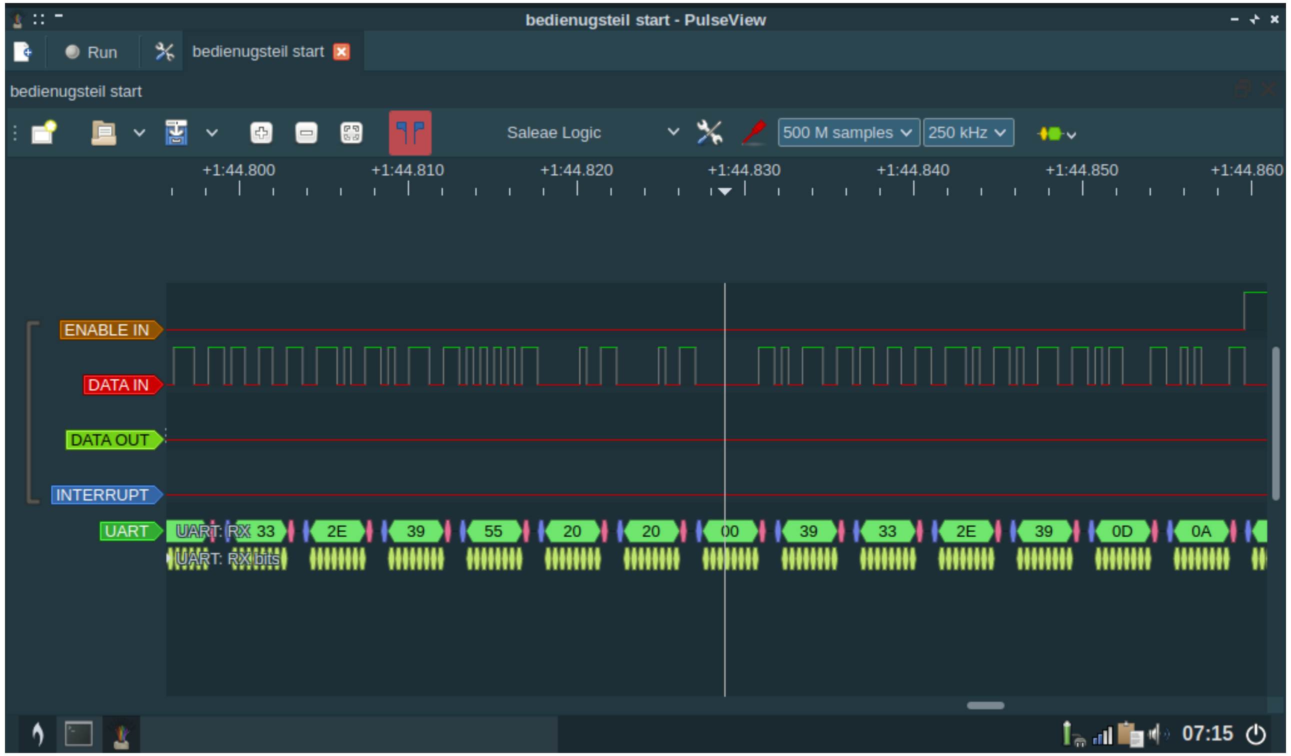
Operator panel communication

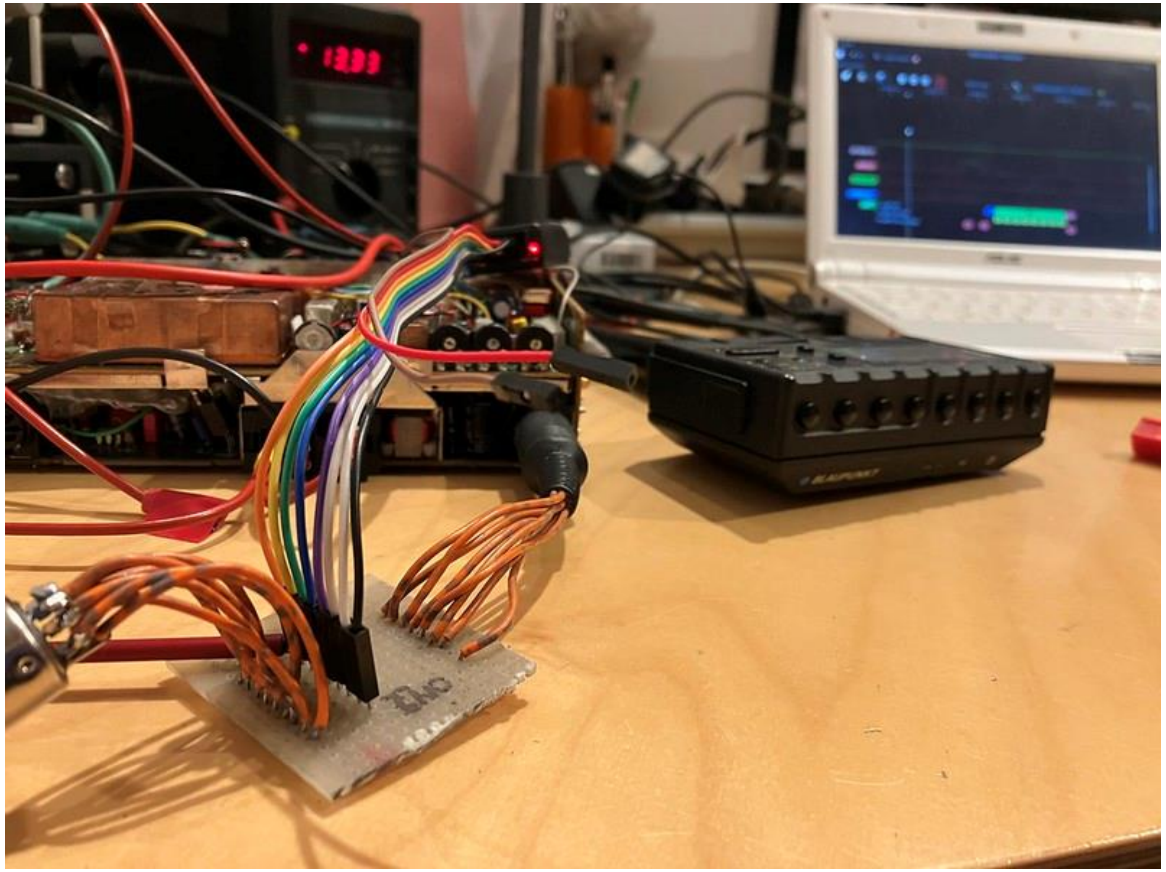
2023-12-17

I built a connector to tap off the data between the operator panel and the main unit. The interface is indeed normal UART both ways, running at 2400 baud, 8 bits, one start bit.

There is data both with the enable signal from the main unit being high and low, but it seems it's low when there are data being sent to the LCD. Perhaps it's high with other data, e.g. LED's? Data to the LCD is ascii by the way.

Keypresses trigger an interrupt (as expected) and some code indicating which key is pressed. I haven't decoded the logic yet.





Data sent to the display right after power-on…
visible UART data converts to this string:
…3.9U <NUL>93.9<CR><LF>…