<u>CB Speech processor – Audio processor/filter</u>

Document version 17-12-2025

Jef Collin

These are some notes on the construction and programming.

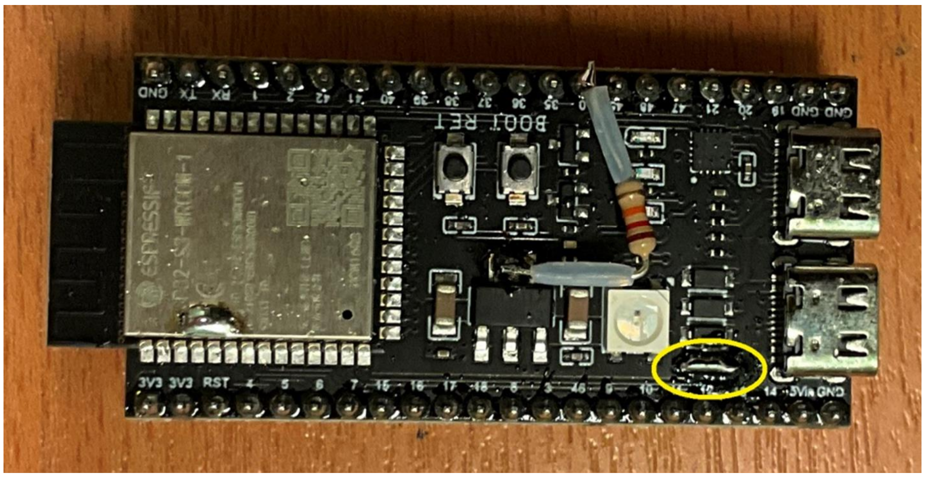
<u>The ESP32-S3 module</u>

As explained in the video the boot button is connected to GPIO0 and has a weak internal pullup, due to an external debounce capacitor this causes the module to go into download mode when the capacitor is not fully charged yet, for example on power on from a fast source or when connecting to the serial monitor after a software reset by the Arduino IDE. To solve this a 10K resistor needs to be added between +3.3V and GPIO0, this creates a stronger pullup allowing the debounce capacitor to charge faster before the reset routine detects it as a low on GPIO0 therefor it will not go into download mode but all other functions such as download from the IDE work as intended.

Adding a resistor to the module is no longer required since the resistor is now included in the PCB design.

Take note where the USB connector goes (for programming), there are two USB ports, one is connected directly to the processor and the other has the usual USB to serial convert IC (we use this one, the top one in the picture).

The project needs a decent 5v power supply, In case you want to test the project with only the USB cable connected short a solder bridge as shown on the picture, this will provide 5V to the 5V pin of the module which is connected to the rest of the circuit.



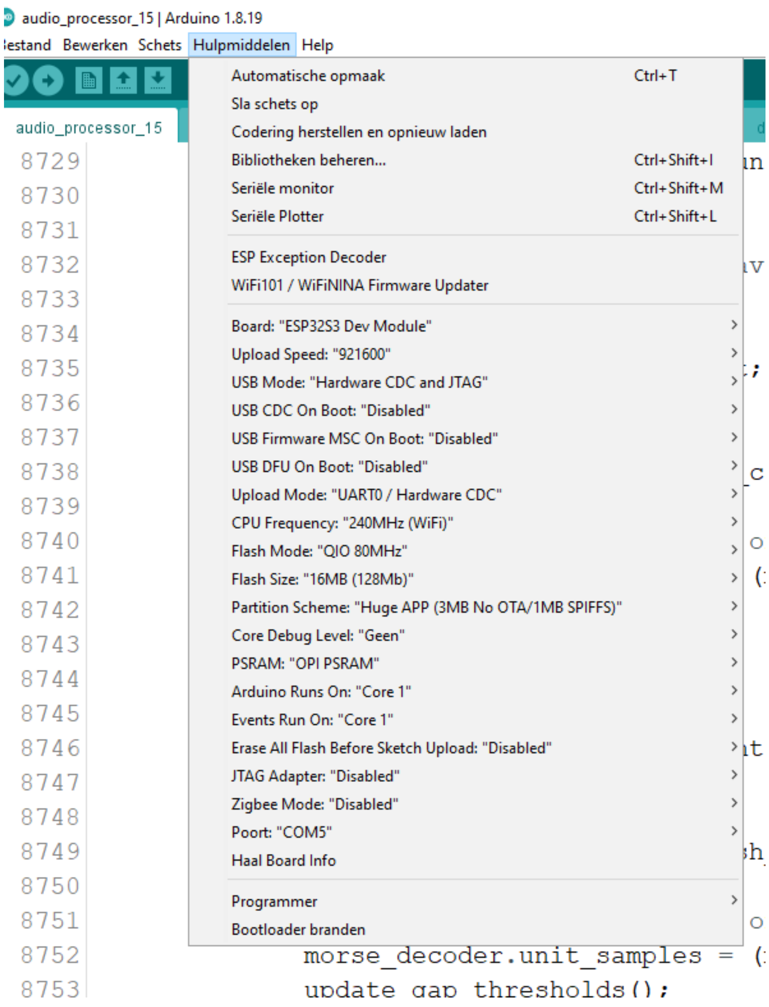
Programming the ESP32-S3

The ESP32 can be programmed in the usual way using the Arduino 1.8.19 development environment.

I strongly advise to use this version of Arduino IDE, install it in portable mode and install the ESP32 add-on libraries, in my modular tester repository you can find a PDF with instructions on how to setup.

Important: both speech processor and audio processor share the same base hence the Arduino code folders can be placed in the same sketchbook folder because the parameters for LVGL are the same.

Select **ESP32S3 Dev Module** as board, make sure the other settings are correct (partition and PSRAM are important settings).



audio_processor_15 | Arduino 1.8.19

Bestand Bewerken Schets Hulpmiddelen Help

audio_processor_15

```
Automatische opmaak                                  Ctrl+T
Sla schets op
Codering herstellen en opnieuw laden
Bibliotheken beheren...                              Ctrl+Shift+I
Seriële monitor                                      Ctrl+Shift+M
Seriële Plotter                                      Ctrl+Shift+L

ESP Exception Decoder
WiFi101 / WiFiNINA Firmware Updater

Board: "ESP32S3 Dev Module"
Upload Speed: "921600"
USB Mode: "Hardware CDC and JTAG"
USB CDC On Boot: "Disabled"
USB Firmware MSC On Boot: "Disabled"
USB DFU On Boot: "Disabled"
Upload Mode: "UART0 / Hardware CDC"
CPU Frequency: "240MHz (WiFi)"
Flash Mode: "QIO 80MHz"
Flash Size: "16MB (128Mb)"
Partition Scheme: "Huge APP (3MB No OTA/1MB SPIFFS)"
Core Debug Level: "Geen"
PSRAM: "OPI PSRAM"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Erase All Flash Before Sketch Upload: "Disabled"
JTAG Adapter: "Disabled"
Zigbee Mode: "Disabled"
Poort: "COM5"
Haal Board Info

Programmer
Bootloader branden
```

```
      morse_decoder.unit_samples = (
      update gap thresholds();
```

Following libraries have to be installed, use the same version number to avoid compilation errors due to breaking changes between library versions. Once it compiles without errors you might try to upgrade the libraries. You can install them via the library manager, some that are not available you can find on github.

LVGL 9.4.0

SigmaDSP 1.1.6

arduinoFFT 2020

LovyanGFX 1.2.9

SD (sparkfun)

Note on LVGL, a configuration include file lv_conf.h is provided, you need to copy that to the portable->sketchbook-> libraries folder (not to the lvgl folder itself). This contains configuration settings for LVGL 9.4.0, if you use another version you need to copy the template from the library folder, change the name to lv_conf.h and edit it, use the old version and compare, very important settings are:

#if 1 /* Set this to "1" to enable content */ (don't ask why but default it's on 0 and you get lots of errors)

#define LV_MEM_SIZE (70 * 1024U)   /**< [bytes] */ (this defines how much RAM space is reserved, too low it will hang during screen refresh or not compile, too high it will reduce the space for variables and cause compilation or other errors)

#define LV_FONT_MONTSERRAT_8 0

You need to set <u>all</u> the Montserrat fonts that are used by the program to 1.

Compilation of the program will take some time the first time, when compiling again with just code changes in the main program it will be shorter due to caching of compiled libraries.

When uploading make sure Serial Monitor and Serial Plotter windows are closed otherwise the ESP32-S3 could go into download mode again after downloading and not perform any EEPROM update (display will be blank).

After programming observe the display for any instructions, wait a moment and then power cycle the speech or audio processor since the DSP has no reset pin exposed and no software reset command.

The code includes the programming code for the DSP board and will program it if it's version number differs from the version number in the Arduino code.

Make sure jumpers are set to SCL, SDA and WP (default setting to program and operate).

See below for the generation of new DSP code.

DSP module ADAU1701/1401

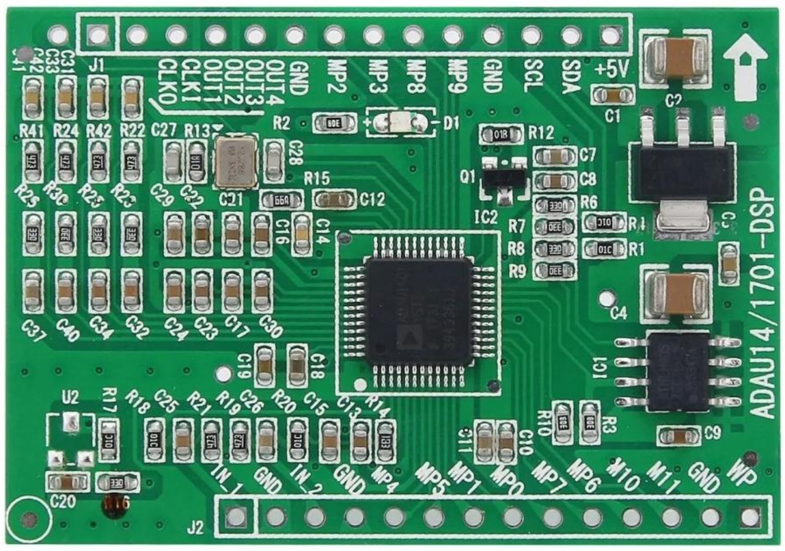
These board are often sold as “learning boards” and can have either ADAU1701 or ADAU1401, both chips are basically the same but different temperature range. Prices may vary and fluctuate due to dollar exchange, I paid around 15 EUR average.

The board contains the DSP, clock oscillator, 5V to 3.3V voltage regulator and EEPROM. On reset the DSP is configured to load its configuration from the EEPROM.

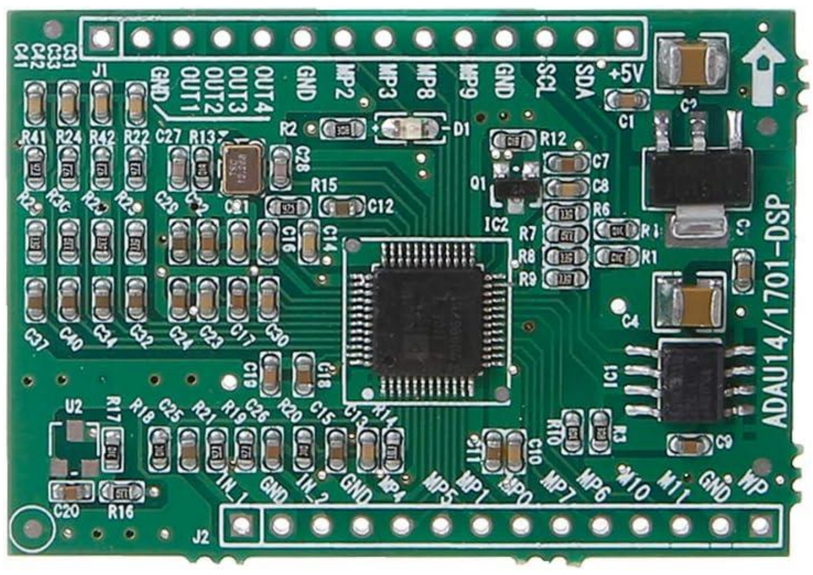
The EEPROM and DSP can be addressed via the I2C bus, programming can be done either using a dedicated programming tool or by the ESP32.

There are several versions of this DSP module, the new version has an extra pin on J1, the lazy designers removed GND from pin 1 and connected clock and inverted clock to this pin and the new pin, the new pin is outside the silkscreen, let’s call it pin -1. My PCB’s are designed to work with the new version (pin 1 not grounded and both 1 and -1 floating) but would probably work with the old version since there are multiple GND pins.

New version (pin 1 and new pin -1 are clock outputs), works.



Old version (pin 1 of J1 is ground), not tested, will probably work.



There are two ways to program the DSP.

Programming from the ESP32

Set jumpers to SCL, SDA and WP, this connects the I2C bus and EEPROM write protect directly to the ESP32, this is also the default setting for operation.

The ESP32 code contains an include file with the program codes for the DSP, a version number is stored in the DSP, on startup that number is read and if different new code is programmed into the EEPROM. A message will appear on the display during and after programming, when done, power cycle.

If changes in DSP configuration are required they can be done in the Sigmastudio environment without the DSP connected, a new include file for the ESP32 can be made, then compile and download into the ESP32 which will update the DSP code. Make sure to update the version number in the ESP32 code to trigger the update (exact number does not matter, it’s just a number to distinguish versions).

Prepare the DSP file to include in the ESP32 code

This is already done but in case you want to change the DSP code you need to follow these steps.

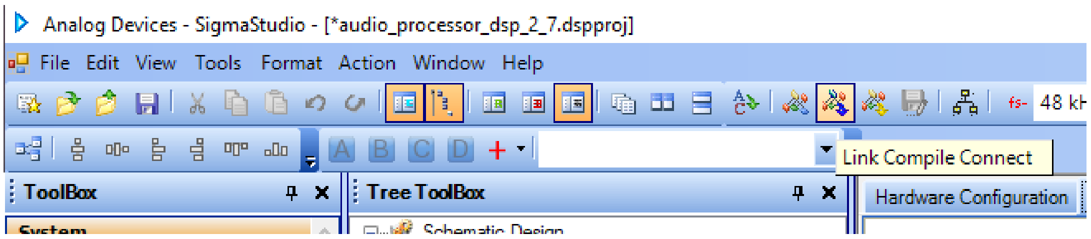
Inside the ESP32 source code folder there is a folder with the DSP project, in that an export folder and files generated by Sigmastudio. Also a powershell script to convert the generated files (from the export folder) into the include file required by the ESP32, the resulting file is called SigmaDSP_parameters.h, this contains the addresses for the building blocks so the ESP32 knows how to find for example the input gain slider and adjust it, plus the DSP configuration code to be stored in the EEPROM of the DSP module.

Open the latest .dspproj file in Sigmastudio and make any changes you like to make, when saving make sure you save it in the correct folder, it is good practice to open a project file and immediately use **Save As** with a different filename and a version number included and make changes in that new copy so you can easily revert back to a previous version.

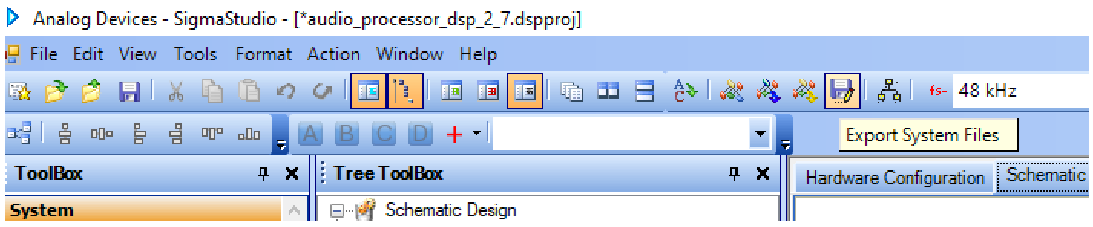
Make sure you don't save the project to the **export** folder.

Delete all files in the **export** folder, these will be recreated. You will get errors if you have older versions output in the export folder when running the PowerShell script.

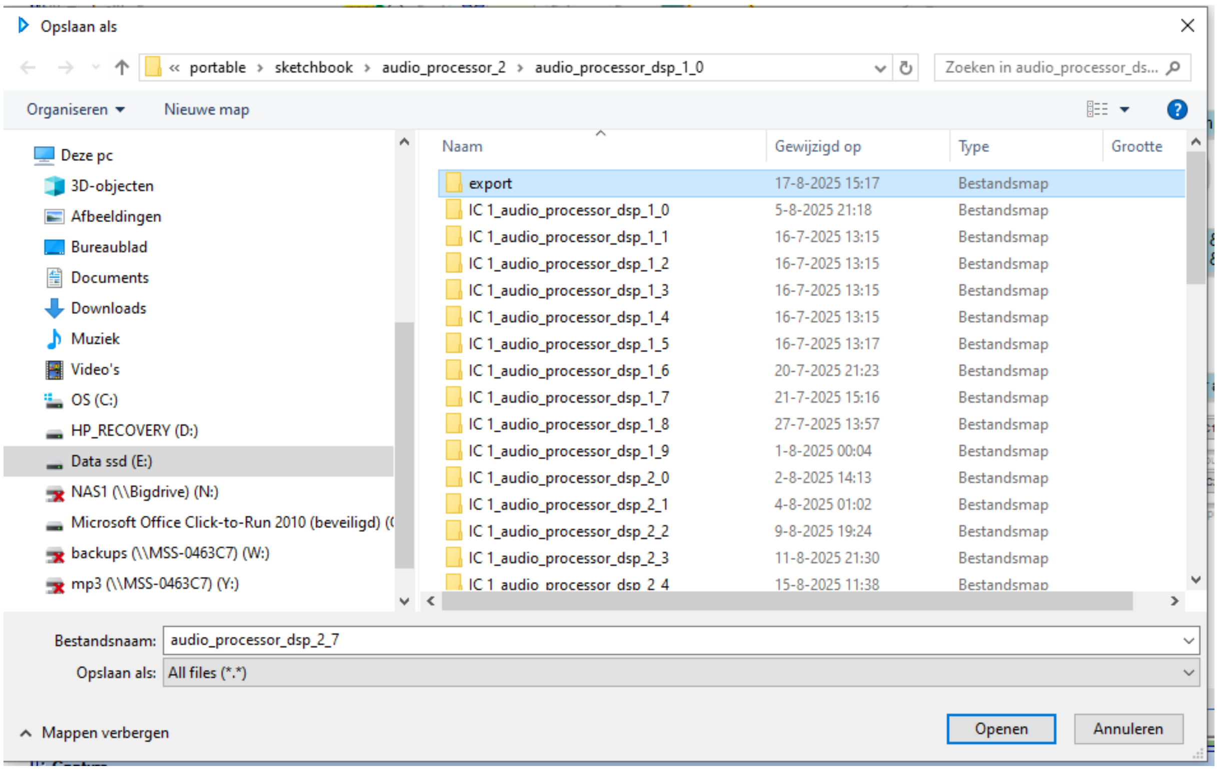
Click **Link Compile Connect**, if any errors occur a window will open.


Analog Devices - SigmaStudio - [*audio_processor_dsp_2_7.dspproj]
File Edit View Tools Format Action Window Help
48 kH
Link Compile Connect
ToolBox
Tree ToolBox
Hardware Configuration
System
Schematic Design

When ready the **Export System Files** button will be enabled, click it.


Analog Devices - SigmaStudio - [*audio_processor_dsp_2_7.dspproj]
File Edit View Tools Format Action Window Help
48 kHz
Export System Files
ToolBox
Tree ToolBox
Hardware Configuration
Schematic
System
Schematic Design

Select the **export** folder and click **Open**.

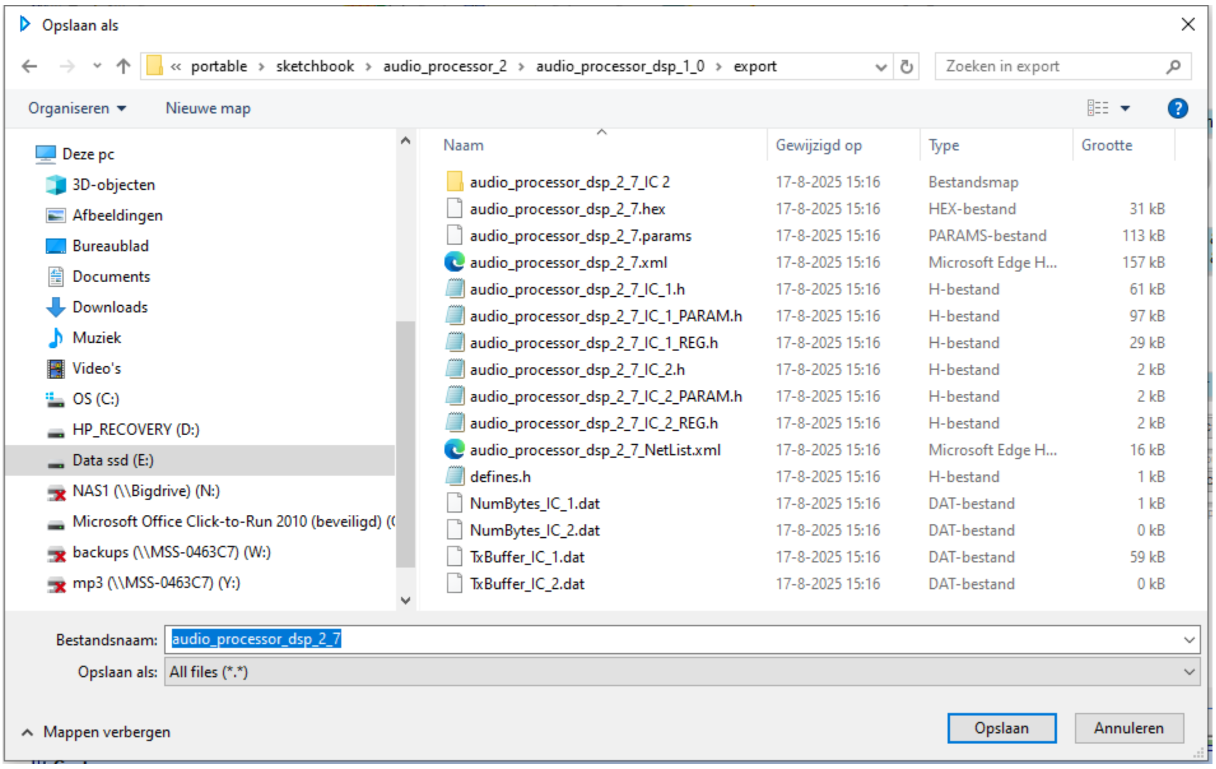


Naam	Gewijzigd op	Type	Grootte
export	17-8-2025 15:17	Bestandsmap	
IC 1_audio_processor_dsp_1_0	5-8-2025 21:18	Bestandsmap	
IC 1_audio_processor_dsp_1_1	16-7-2025 13:15	Bestandsmap	
IC 1_audio_processor_dsp_1_2	16-7-2025 13:15	Bestandsmap	
IC 1_audio_processor_dsp_1_3	16-7-2025 13:15	Bestandsmap	
IC 1_audio_processor_dsp_1_4	16-7-2025 13:15	Bestandsmap	
IC 1_audio_processor_dsp_1_5	16-7-2025 13:17	Bestandsmap	
IC 1_audio_processor_dsp_1_6	20-7-2025 21:23	Bestandsmap	
IC 1_audio_processor_dsp_1_7	21-7-2025 15:16	Bestandsmap	
IC 1_audio_processor_dsp_1_8	27-7-2025 13:57	Bestandsmap	
IC 1_audio_processor_dsp_1_9	1-8-2025 00:04	Bestandsmap	
IC 1_audio_processor_dsp_2_0	2-8-2025 14:13	Bestandsmap	
IC 1_audio_processor_dsp_2_1	4-8-2025 01:02	Bestandsmap	
IC 1_audio_processor_dsp_2_2	9-8-2025 19:24	Bestandsmap	
IC 1_audio_processor_dsp_2_3	11-8-2025 21:30	Bestandsmap	
IC 1_audio_processor_dsp_2_4	15-8-2025 11:38	Bestandsmap	

Bestandsnaam: audio_processor_dsp_2_7

Opslaan als: All files (*.*)

Click **Save** to export the system files to the **export** folder.

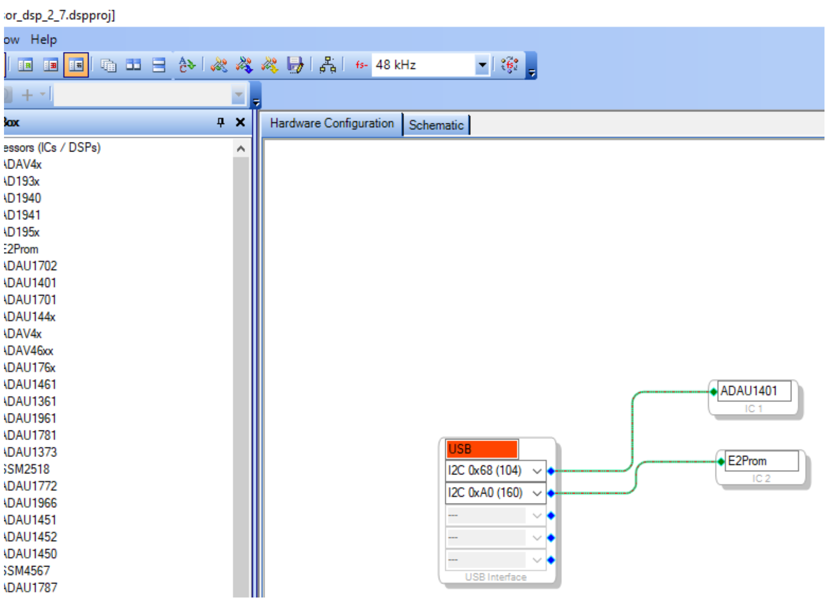


Naam	Gewijzigd op	Type	Grootte
audio_processor_dsp_2_7_IC 2	17-8-2025 15:16	Bestandsmap	
audio_processor_dsp_2_7.hex	17-8-2025 15:16	HEX-bestand	31 kB
audio_processor_dsp_2_7.params	17-8-2025 15:16	PARAMS-bestand	113 kB
audio_processor_dsp_2_7.xml	17-8-2025 15:16	Microsoft Edge H...	157 kB
audio_processor_dsp_2_7_IC_1.h	17-8-2025 15:16	H-bestand	61 kB
audio_processor_dsp_2_7_IC_1_PARAM.h	17-8-2025 15:16	H-bestand	97 kB
audio_processor_dsp_2_7_IC_1_REG.h	17-8-2025 15:16	H-bestand	29 kB
audio_processor_dsp_2_7_IC_2.h	17-8-2025 15:16	H-bestand	2 kB
audio_processor_dsp_2_7_IC_2_PARAM.h	17-8-2025 15:16	H-bestand	2 kB
audio_processor_dsp_2_7_IC_2_REG.h	17-8-2025 15:16	H-bestand	2 kB
audio_processor_dsp_2_7_NetList.xml	17-8-2025 15:16	Microsoft Edge H...	16 kB
defines.h	17-8-2025 15:16	H-bestand	1 kB
NumBytes_IC_1.dat	17-8-2025 15:16	DAT-bestand	1 kB
NumBytes_IC_2.dat	17-8-2025 15:16	DAT-bestand	0 kB
TxBuffer_IC_1.dat	17-8-2025 15:16	DAT-bestand	59 kB
TxBuffer_IC_2.dat	17-8-2025 15:16	DAT-bestand	0 kB

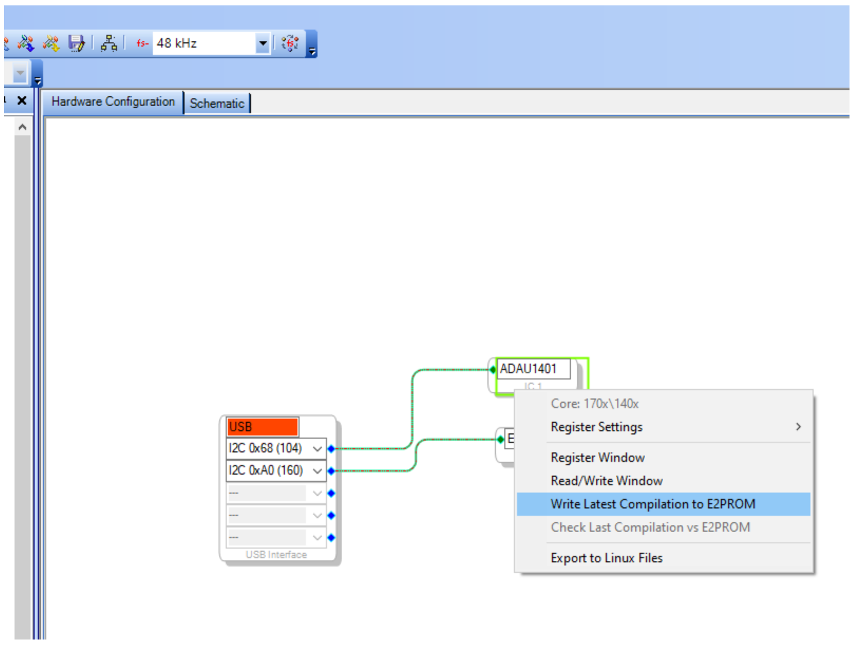
Bestandsnaam: audio_processor_dsp_2_7

Opslaan als: All files (*.*)

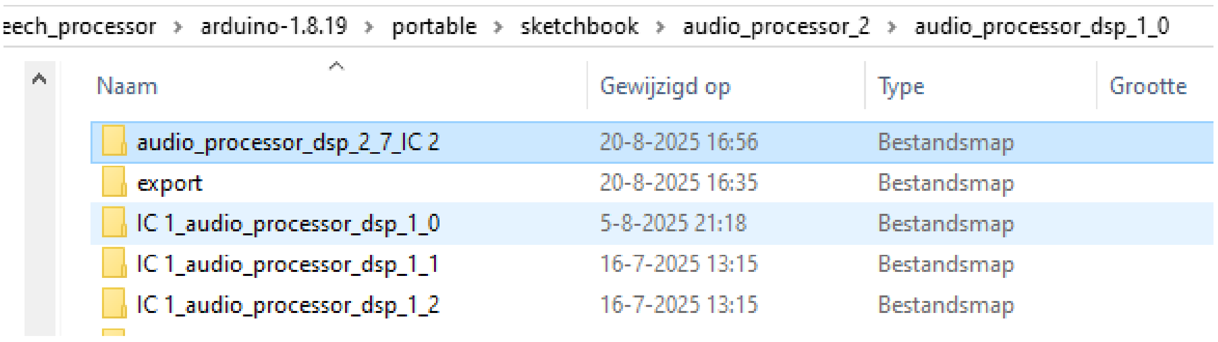
Select the **Hardware Configuration** tab.



Put the cursor on the text **IC1** (not on the ADAU1401 field**)**, right click and select **Write Latest Compilation to E2PROM**. Since the DSP is not connected it can't program it but it will create the EEPROM files we need to include in our code.



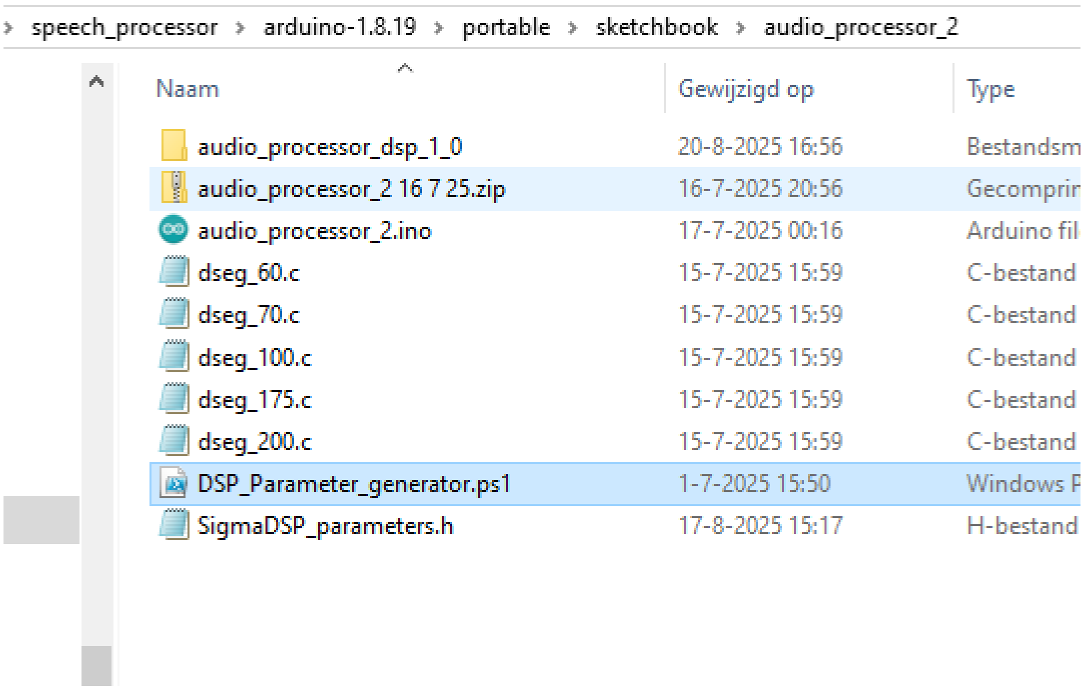
When done a folder is created as in this example ending in …_IC 2. Drag this folder into the export folder. Make sure there are no left over …_IC 2 folders from older projects in this project folder or the export folder, delete if any.



eech_processor › arduino-1.8.19 › portable › sketchbook › audio_processor_2 › audio_processor_dsp_1_0

Naam	Gewijzigd op	Type	Grootte
audio_processor_dsp_2_7_IC 2	20-8-2025 16:56	Bestandsmap	
export	20-8-2025 16:35	Bestandsmap	
IC 1_audio_processor_dsp_1_0	5-8-2025 21:18	Bestandsmap	
IC 1_audio_processor_dsp_1_1	16-7-2025 13:15	Bestandsmap	
IC 1_audio_processor_dsp_1_2	16-7-2025 13:15	Bestandsmap	

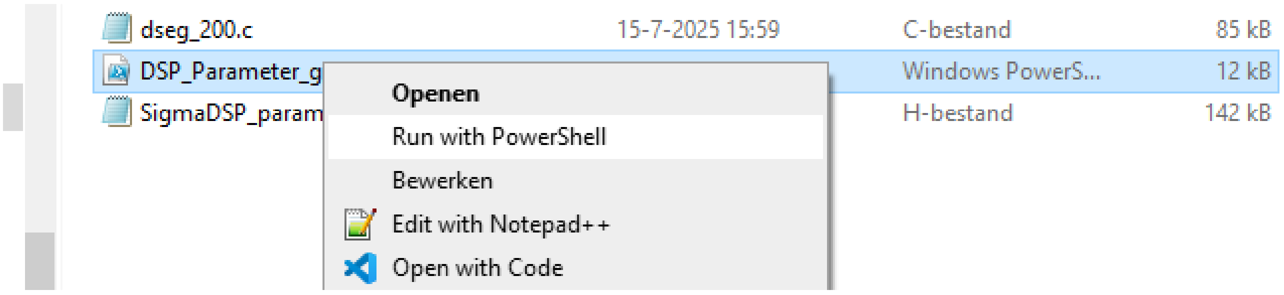
Go to the Arduino sketch folder, make sure the Arduino IDE is closed otherwise the IDE will not detect the changes and uses a cached include file.



› speech_processor › arduino-1.8.19 › portable › sketchbook › audio_processor_2

Naam	Gewijzigd op	Type
audio_processor_dsp_1_0	20-8-2025 16:56	Bestandsm
audio_processor_2 16 7 25.zip	16-7-2025 20:56	Gecomprir
audio_processor_2.ino	17-7-2025 00:16	Arduino fil
dseg_60.c	15-7-2025 15:59	C-bestand
dseg_70.c	15-7-2025 15:59	C-bestand
dseg_100.c	15-7-2025 15:59	C-bestand
dseg_175.c	15-7-2025 15:59	C-bestand
dseg_200.c	15-7-2025 15:59	C-bestand
DSP_Parameter_generator.ps1	1-7-2025 15:50	Windows P
SigmaDSP_parameters.h	17-8-2025 15:17	H-bestand

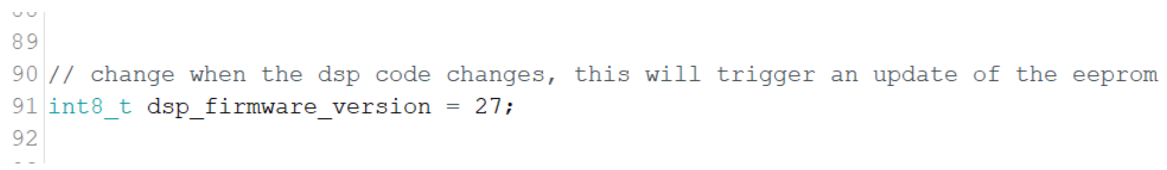
Rightclick **DSP_Parameter_generator.ps1** and select **Run with PowerShell**, if no errors occur a file **SigmaDSP_parameters.h** is created or updated.



dseg_200.c	15-7-2025 15:59	C-bestand	85 kB
DSP_Parameter_g		Windows PowerS...	12 kB
SigmaDSP_param		H-bestand	142 kB

This file contains not only the EEPROM code for the DSP but also the mapping of all element addresses that are used by the ESP to change the parameters of filters, equalizers, amplifiers and so on, it is important that DSP code – include file - ESP code are all in sync.

Open the Arduino IDE and change the version number dsp_firmware_version (0-127), the exact number does not matter much because it's just used to detect a difference in version triggering the update of the EEPROM, an update can be forced by changing the version number.



```

// change when the dsp code changes, this will trigger an update of the eeprom
int8_t dsp_firmware_version = 27;

```

Compile and download the program into the ESP, after downloading it will detect the change in version number, write the DSP code to its EEPROM and display a power cycle message.

Programming from Sigmastudio

This method can be used to test new configurations and/or interact with the DSP, you can make live changes to multiplexers, sliders, amplifier gain, filter parameters.

However there is no version number written by Sigmastudio hence the ESP32 code will probably detect a change in version number and overwrite the code with the code in its include file so make sure you prepare new files if changes are made to the DSP program.

Using the download button loads the code into the DSP's RAM and not the EEPROM so on power up the code is lost, programming the EEPROM is an additional step.

Very important, all elements in the DSP configuration are referenced by identifier names and addresses via the include file, replacing elements breaks this link and results in compilation errors in the ESP code, adding elements keeps the references to the existing elements but their addresses will change, this is not a breaking change and taken care of by a new include file.

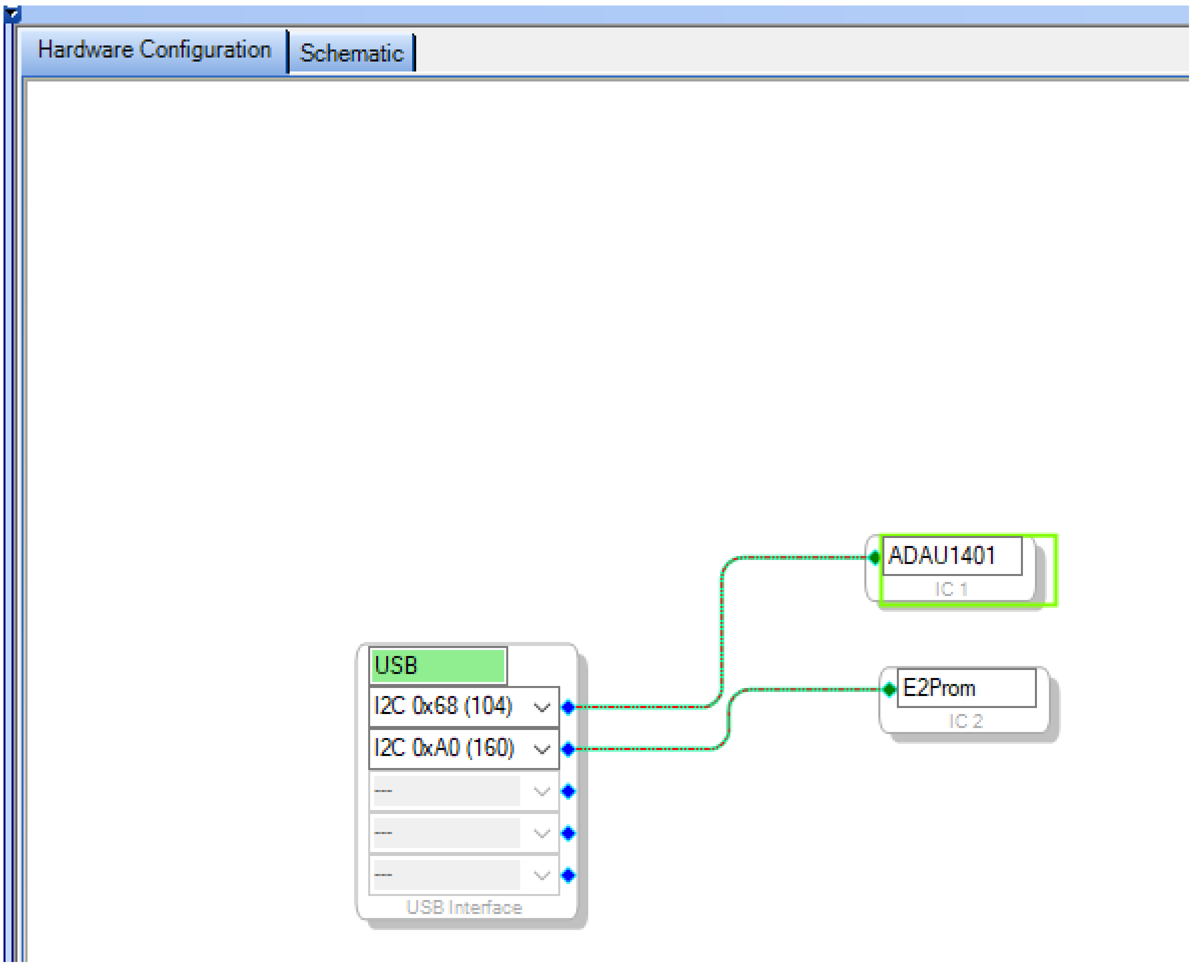
About the programming tool, a cheap cypress CY7C68013A board is used, to prepare this board as a programming tool and the installation of freeusbi, refer to instructions on the net.

Note on using freeusbi, the cypress board is sensitive to its power so power it as direct as possible, no usb monitors or switches in between, works best from a powered hub.

The DSP connects to the programming tool, remove all jumpers and connect GND, SCL, SDA to the tool.

Power on the DSP and tool, check in the hardware configuration tab if USB and the DSP show up in green.

I will skip the instructions to program the EEPROM.


Hardware Configuration
Schematic
USB
I2C 0x68 (104)
I2C 0xA0 (160)

USB Interface
ADAU1401
IC 1
E2Prom
IC 2

Some random build notes

A lot is explained in the videos.

Mount PCB with serrated M3 washers and metal screws to ensure optimal contact between ground of PCB and metal housing (the nuts are a little larger than the exposed pad therefor they are on the solder mask and do not make contact with the ground planes). No separate ground wire is required between housing and power supply ground.

If this PCB to housing ground contact is not optimal the audio path will pick up the display access noise resulting in a low frequency clicking noise especially where frequent screen updates are done like the scope and spectrum screen and screens with VU meter.

Only use isolated 3.5mm (plastic) jacks, some common pins are not connected to ground and for others it might create a ground loop.

The speech processor output is floating due to the 600:600 ohm isolation transformer, in some cases there might be a little hum in the modulation so we need to ground somewhere, inside the GX16 plug that goes into the speech processor I added a bare wire to the common pin of the microphone output and clamped it with the bracket of the plug so it is connected to the metal part of the plug and by connecting to the housing of the speech processor.

The PTT output of the speech processor is intended for an optocoupler on the other side as implemented in the audio processor. This allows the audio processor to mute during transmission.

The audio processor, on some cb radios the speaker output is not referred fixed to ground, connecting it to the audio processor completes a ground loop and you will get the tx sound and its very loud. I installed a 600:600 transformer in the cable between speaker output and speaker input to provide isolation.

The headphone amplifier is a standard MAX98357A module that plugs into the PCB. On the speech processor there is an optional connection below the DSP module to connect to an analog amplifier.