The OST Morse Box DG with Decoder and Generator

... AND WITH

Iambic A & B!

Ostend Radio Club UBA-OST V2.12

Save some paper and view this manual on your computer!

Contents

THE OST MORSEBOX DG – V2.12

Description - Purpose

This expansion of the **OST Morse Box**, which was developed in 2020, remains fully compatible with the existing hardware. So you don't have to build a new Morse Box!
In this manual we only describe what new functions have been added, for the rest we refer to the basic manual (see https://github.com/on7dq/OST-Morse-Box).

What's new in this version?

Decoder

The new firmware decodes all CW that you transmit and shows it on the display. To decode the received CW, you'll need a simple extra hardware decoder. Normally, you will take the audio from the VHF/UHF transceiver, but one could also provide an input from another source (e.g. HF transceiver). The speed (WPM) of the code is shown on the display, and all output is also sent to the Serial Monitor of the Arduino IDE or the Windows Program.

The hardware decoder is a small circuit board that is easily connected to the basic Morse Box, with only 4 wires. Two versions have been developed:
- the **PLL Decoder** with the well-known LM567, built with 'through-hole' components
- the **BPF decoder** based on a band pass filter and level detector, in an SMD version.

All necessary files to make the extra circuit boards are on github*, but you could easily make the LM567 circuit on a piece of perfboard (see my example further on).

AF Generator

As the routines to generate a pure sine wave were already present in the first version of the OST Morse Box, it was only a small step to add a basic AF generator.
This can be used to align the decoder(s), or as a test tone in the shack ...
The AF frequency ranges from 50 Hz to 2000 Hz. The level of the output signal depends on the frequency, this is due to the low pass filter on the PWM output of the Arduino.

Iambic Keyer

The basis Morse Box had only the "plain iambic" keying mode, without a dot/dash memory.
This version adds the more commonly used modes **Iambic A** and **Iambic B.**

Software

All software was brought to version 2.12 for this extension, so that there is some unity in the numbers.
The Windows program contains a number of new functions.
And with the XLOADER program, it's now even easier to update the Arduino firmware.

On the cover of this manual is the OST Morse Box DG that I built in August 2021. It contains all possible 'bells & whistles' the project has to offer. More details are on my blog**.
Of course, most of this new development is thanks to club member **Gil, ONL12523**.
Have fun with this expanded version … The OST Morse Box DG!

* https://github.com/on7dq/OST-Morse-Box-DG **Luc, ON7DQ – August 2021**
** https://on7dq.blogspot.com/2021/08/ost-morse-box-dg-on7dq-version.html

The Hardware

Remark: better resolution circuit and placement diagrams are at the end of this manual.

Both decoders are connected in the same way via a 5-pin header (**Head1**).
If you want to build both decoders (e.g. to compare their results), you can easily interchange them.
The connections are:

1. LED : cathode of an LED, connect the anode to the +5V rail (this LED is optional)
2. OUT : connect to Arduino D2 (available on the 'Paddle Test' jumper J9)
3. +5V : connect to +5V in the Morse Box, e.g. near the Jumper Block
4. GND : connect to the GND of the Morse Box (ditto)
5. IN: the AF signal of the transceiver (see below)

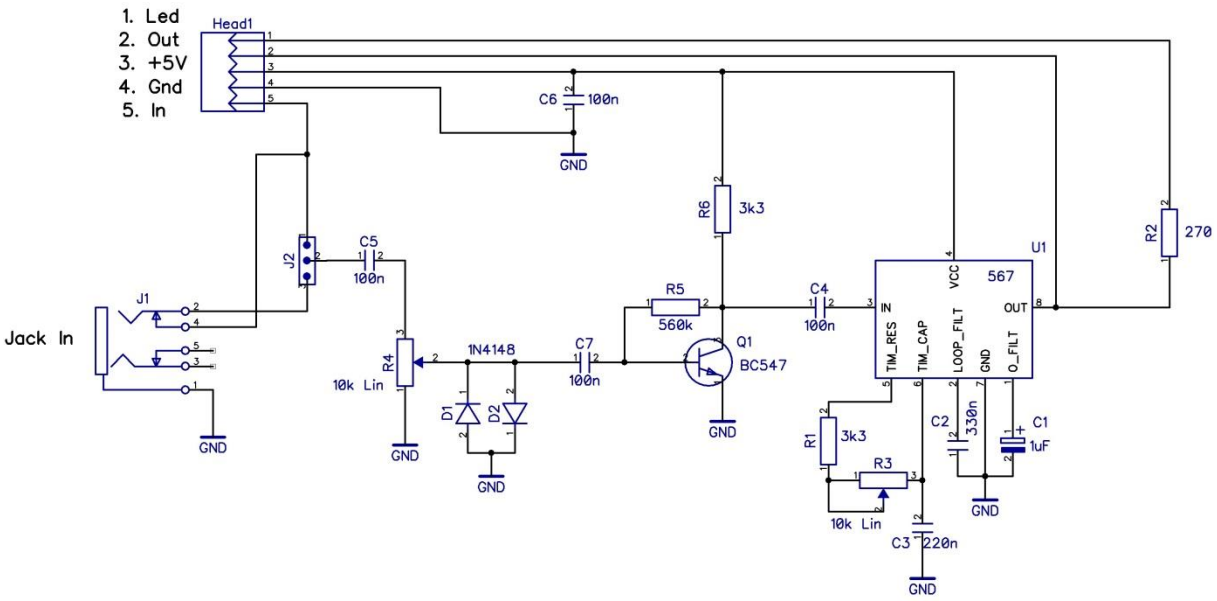
PLL Decoder with LM567

The circuit with the LM567 is used in most decoders that you can find on the internet.
I based mine on the design of the Veron [1].
An audio preamp has been added, but it is optional, see remark below the Parts List.

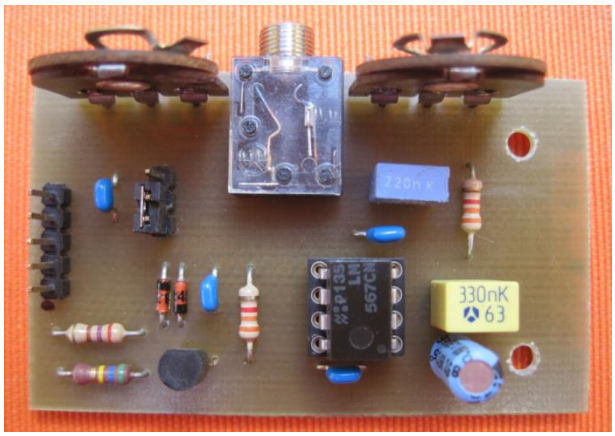
This decoder has a 3.5 mm jack to connect an external AF signal.
At jumper J2 you can connect a single pole toggle switch, or you just put a jumper on pins 2-3 of J2, in which case the signal is switched over by plugging in the cable into J1 ('Jack In').

You probably won't adjust potentiometers R3 and R4 very often, so you could use some large model trimmers instead, and even mount the whole assembly at the back of your Morse Box if you wish.

Circuit Diagram



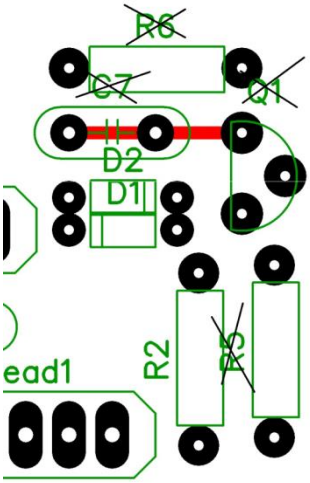
Board



Parts List

Name	Type	Value
C1	Elco	1 uF
C2	Cap	330 nF
C3	Cap	220 nF
C4, C5, C6	Cap	100 nF
D1, D2	Diode	1N4148
Head1	Header 5 pins	
J1	Jack 3,5 mm	
J2	Header 3 pins	
R1	R 1/4 Watt	3k3
R2	R 1/4 Watt	270
R3, R4	Potentiometer or Trimmer	10k Linear
U1	Tone Decoder	LM567CN
Q1	Transistor	BC547B
R5	R 1/4 Watt	560k
R6	R 1/4 Watt	3k3
C7	Cap	100 nF

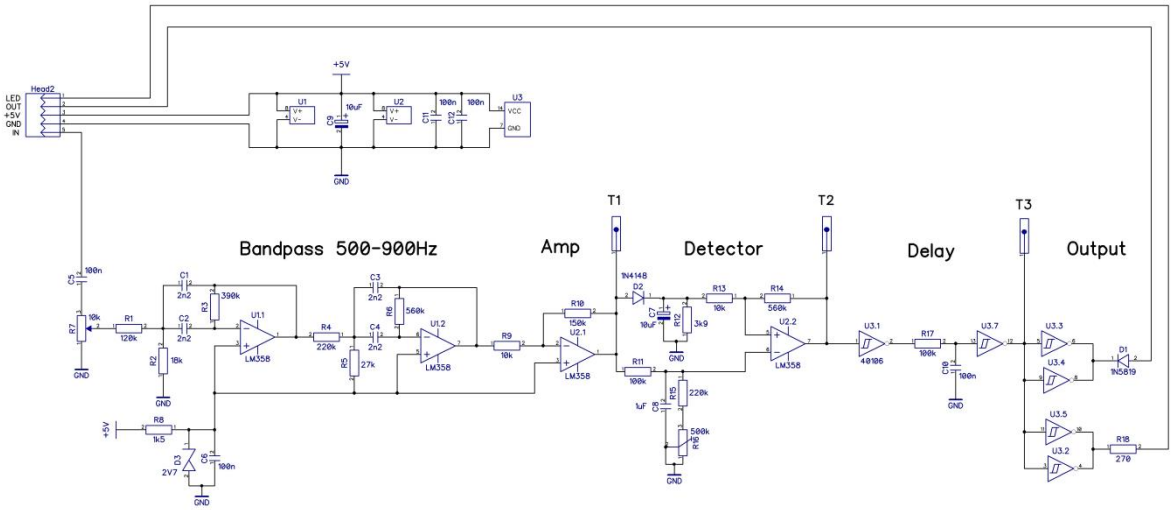
In case you don't need the preamplifier, you can omit the components in RED, and place a jumper wire like the red line in this drawing.



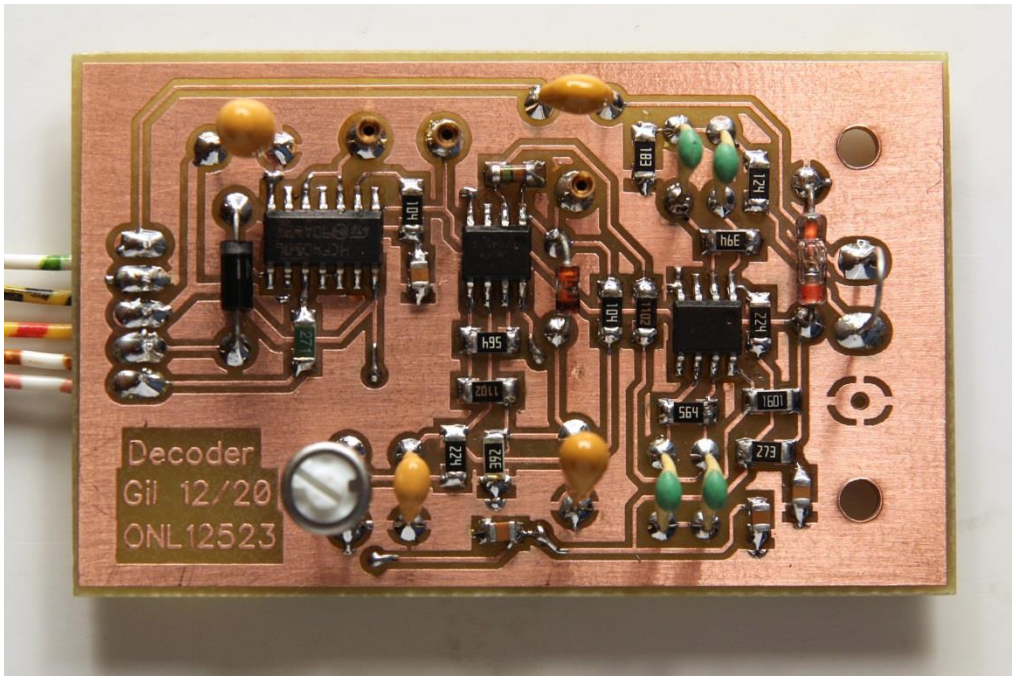
BPF Decoder with bandpass filter and level detector

Gil took a different approach, and built this decoder circuit, based on a design that was once discussed on the Elektor Labs website. [2]
This board has no input for an external signal, but you can always make one yourself by interrupting the line from **In** to **C5**.

Circuit Diagram



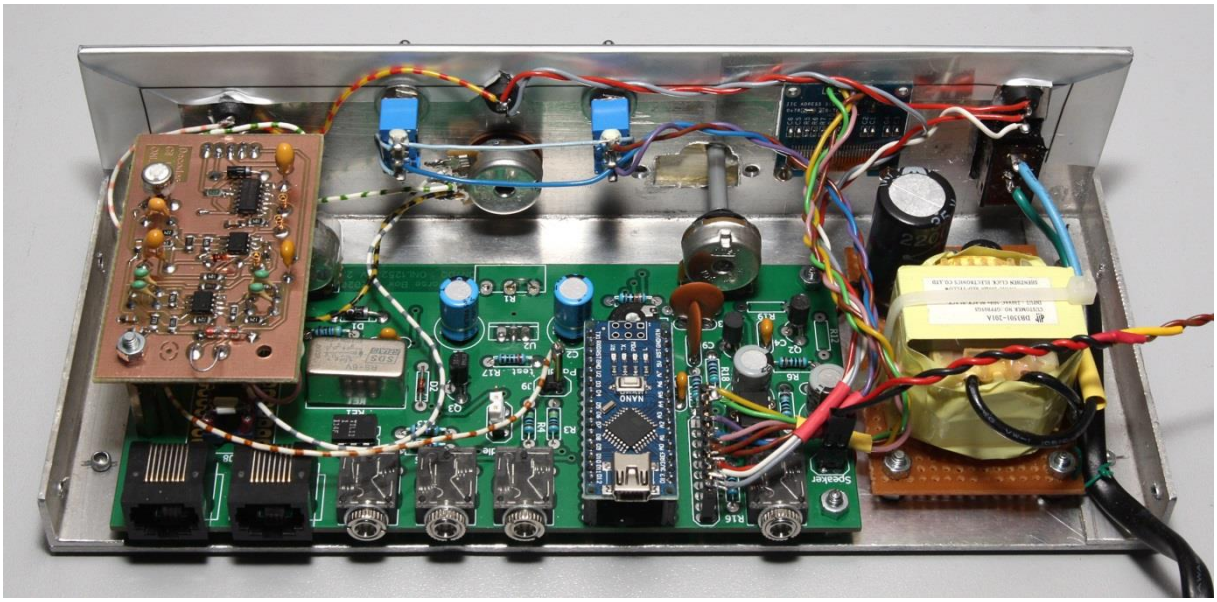
Board



Parts List

Name	Value	Model	Pattern
C1	2n2	Cera P5.08	Cera P5.08
C2	2n2	Cera P5.08	Cera P5.08
C3	2n2	Cera P5.08	Cera P5.08
C4	2n2	Cera P5.08	Cera P5.08
C5	100n	Cera P5.08	Cera P5.08
C6	100n	Cap SMD 0805	CAP_0805
C7	10uF	Elec re P5,04/D	10 Elec re P5,04/D10
C8	1uF	Cera P5.08	Cera P5.08
C9	10uF	Elek re P5,04/D	13 Elek re P5,04/D13
C10	100n	Cap SMD 0805	CAP_0805
C11	100n	Cap SMD 0805	CAP_0805
C12	100n	Cap SMD 0805	CAP_0805
D1	1N581	9 Diode P10	Diode P10
D2	1N414	8 Diode P7,5	Diode P7,5
D3	2V7	DIO_ZENER_4	Diode P10
Head2		Header 5Pins	Header 5Pins
Pads3		Pad 1	D3 Pad 1D3
Pads4		Pad 1	D3 Pad 1D3
Pads5		Pad 1	D3 Pad 1D3
R1	120k	Res SMD 1206	RES_1206
R2	18k	Res SMD 1206	RES_1206
R3	390k	Res SMD 1206	RES_1206
R4	220k	Res SMD 1206	RES_1206
R5	27k	Res SMD 1206	RES_1206
R6	560k	Res SMD 1206	RES_1206
R7	10k	R potentiometer	Potentiometer
R8	1k5	Res SMD 1206	RES_1206
R9	10k	Res SMD 1206	RES_1206
R10	150k	Res SMD 1206	RES_1206
R11	100k	Res SMD 1206	RES_1206
R12	3k9	Res SMD 1206	RES_1206
R13	10k	Res SMD 1206	RES_1206
R14	560k	Res SMD 1206	RES_1206
R15	220k	Res SMD 1206	RES_1206
R16	500k	Trimmer rond	Trimmer rond
R17	100k	Res SMD 1206	RES_1206
R18	270	Res SMD 1206	RES_1206
U1	LM358	LM358D	SOIC-8/150mil
U2	LM358	LM358D	SOIC-8/150mil
U3	40106	HCF40106BM1	SOIC-14/10mil

This is how Gil built the BPF decoder into his existing hardware. He repurposed the DELAY potmeter into a level control for the decoder.



Just an idea for the “diehards” … you could build BOTH decoder circuits. Then take the OUT signal of both and use a single pole toggle switch to connect either of them to input D2.
Both decoders can be fed with the same AF signal, or use one for VF/UHF, the other for HF signals.

AF Signal

Now you have to pay some attention …

With certain transceivers you have an 'RX AUDIO' signal on the microphone connector, if not you will have to find an alternative. We’ll give some suggestions here.

Also refer to the basic manual, where the Jumper Block is discussed.

The first example will again be the **Kenwood TM-733E**.
This transceiver has the received audio on pin 2 and with a FIXED level, perfect!
Pin 2 was not connected to anything in the basic manual, so this is an ideal situation: connect pin2 of the 16-pin IC-socket to the input of the decoder, that is pin 5 on **Head1**. Done!

Now **Icom**, as an example again the **IC-706MkIIG.** Here, the received audio is available at pin 3, but this time with a variable level which depends on the position of the volume potentiometer.
So try to always listen at about the same volume level, and adjust the decoder accordingly.
Connect pin 3 of the 16-pin IC socket to the input of the decoder, pin 5 on **Head1**.

ALTERNATIVE for Icom

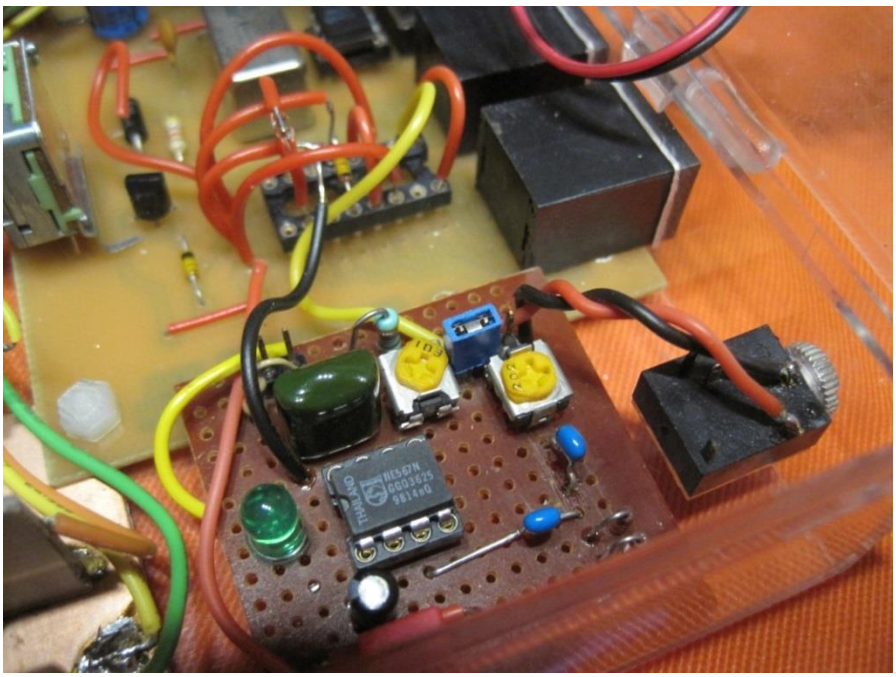
Most Icom transceivers have a 13-pin DIN connector at the back, usually called **ACC** (or sometimes it’s an 8-pin ACC1 + a 7-pin ACC2). Check the appropriate manual to determine which pin provides the fixed-level audio. Make a shielded cable to connect this signal to the input of the decoder.
For the IC-706MkIIG this is pin 12, and GND can be found on pin 2 of the ACC connector.

Not so good news for **Yaesu** … in the **FT-857D**, like in most of their transceivers, there is no audio available on the MIC connector, sorry. The best you can do is to take the audio on the **DATA** connector if you rig has one: GND is pin 2, Audio (= DATA OUT) is pin 5, on the standard 6-pin DIN connector.

ALTERNATIVE
For all other cases; make (or buy) an external speaker for your transceiver, and make a tap at the speaker, e.g. with a 100 Ω resistor in series. Again, the signal level will depend on the position of the volume potentiometer.

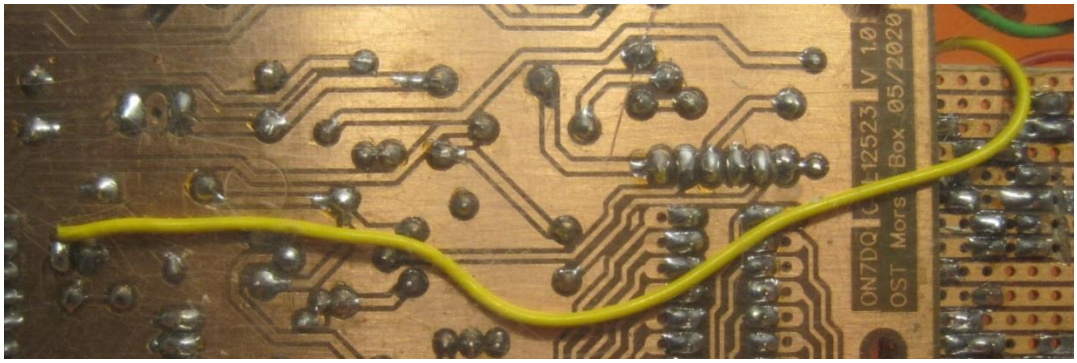
Integrating the decoder into the Morse Box

I built a prototype of the LM567 decoder on a piece of perfboard, and this is how I built it into my 'cigar box':



I didn't use potentiometers, but two small trimmers, and the LED is mounted on the little board.

Three of the four necessary connections were very nearby: +5V, GND and AF IN are all found at the Jumper Block. Only the decoder output needs a longer wire to connect it to pin D2 (= the 'Paddle Test' header). See the long wire which runs under the PCB.



The Software

The Arduino sketch

Morse decoder

The Morse code that you send via the key or paddle is decoded internally and displayed in **lowercase** on the display. For this, you need to do nothing more than load the new firmware into the Arduino (see below). You can also use this decoding locally, to practice your keying, when you don't connect a transceiver, or shut the transceiver down for the duration of the practice.

Received signals have to be decoded with a hardware decoder (see above).

The output of the decoder is sent to input D2 of the Arduino.
In the basic Morse Box, this input was the 'Paddle test' jumper, and it still is.
But now, instead of a jumper, you place a cable with a 2-pin female header. If you still wish to use the paddle test function, place a jumper and reset the Arduino (see basic manual).

The received code is displayed in **uppercase**.

Exactly what is decoded is determined by the state of the PTT.
As long as the PTT is in the active state, the own sent code is shown.
When the PTT is not active, the hardware decoded Morse code (via D2) is shown.
After 15 seconds of inactivity, the decoder resets to 20 WPM and the main screen reappears.

You can disable the decoder if you want. Either you do this in software via an AT command, or via the new button in the Windows program (see below).
However, if you wish to have a direct ON/OFF button for the decoder, then connect an extra push button to connect input A3 to GND. A3 is available on the 10-pin header J6, pin 3.
A pull-up resistor is not needed.

The state of the decoder (ON or OFF) is always stored in EEPROM, so that on a next start or after a reset, you get back the same state.

The push button also has an **extra function**, you can use it to start the generator (see below).
Let's say the the decoder was ON, then the order of operations is:
Decoder OFF > Generator ON > Generator OFF & Decoder again ON > Decoder OFF etc.
If you do not want to use the decoder after using the generator, briefly touch the key or paddle, then the generator goes OFF and the decoder remains also OFF.

Generator

The Morse Box can now also be used as a simple tone generator for adjusting filters and decoders. The frequency can be set with an AT command or is freely adjustable with the WPM potentiometer when starting the generator with the button in the Windows program (or the Decoder pushbutton). Tuning with the WPM potentiometer is not perfect, the frequency may skip a few Hz every now and then. If you want an exact frequency, e.g. 822 Hz, then it's best to use the AT command.

The level of the output signal depends on the frequency, this is due to the low pass filter on the PWM output of the Arduino. If you have a specific use for this generator, you are of course free to disconnect the LPF and connect another filter. This is outside the scope of our Morse Box project however ...

Iambic Modes

The OST Morse Box now has three keying modes: plain Iambic, Iambic mode A and Iambic mode B. The mode can be set in the Windows Program, via File > EEPROM_Settings, after which the selected mode is retained in the Arduino EEEPROM memory. Or you can use the AT-commands, see below.

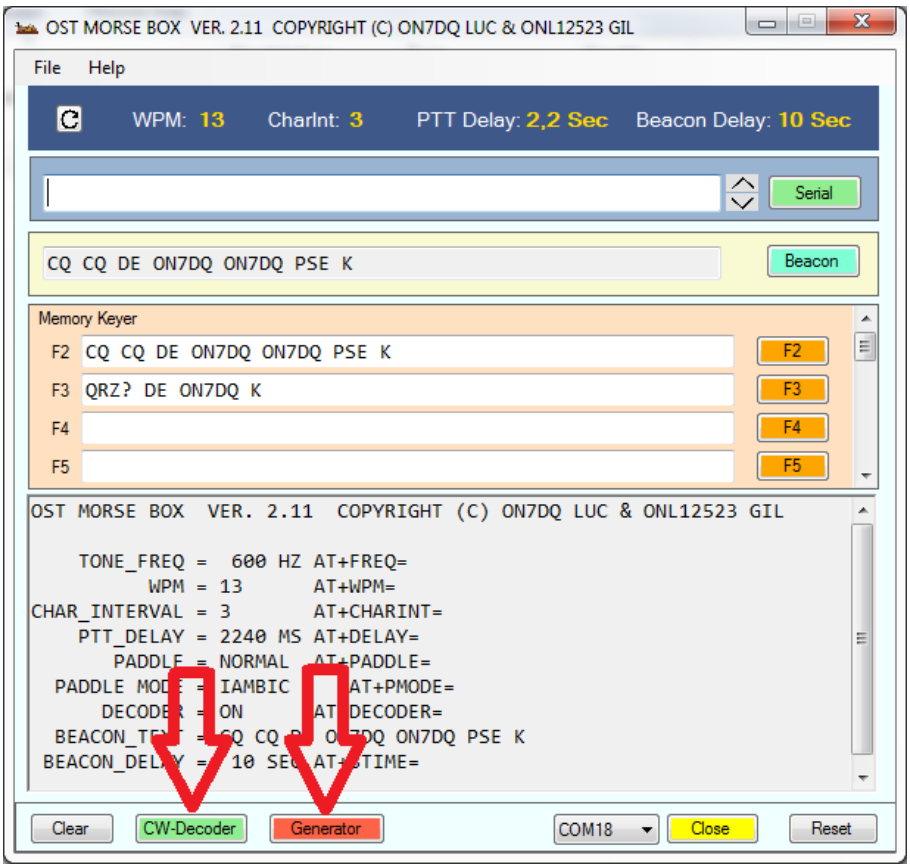
Personal callsign on the start screen

In the Arduino code, an ID or Callsign can be entered at line 14.
It is shown briefly on the display after start-up.

New AT commands

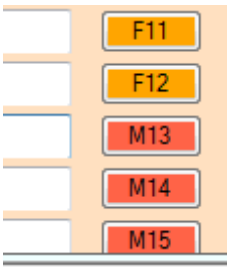
AT+DECODER=ON	Morse decoder ON
AT+DECODER=OFF	Morse decoder OFF
AT+DECODER?	Query decoder status
AT+GEN= 800	Start the generator with a frequency between 50 and 2000 Hz
AT+GEN=0	Stop the generator
AT+GEN=X	Start the generator, frequency settable with WPM potentiometer the range is limited to 450 Hz .. 1450 Hz
AT+GEN?	Query generator frequency
AT+PMODE=0	Plain Iambic
AT+PMODE=1	Iambic A
AT+PMODE=2	Iambic B

Windows program



At the bottom of the main screen, two extra Buttons

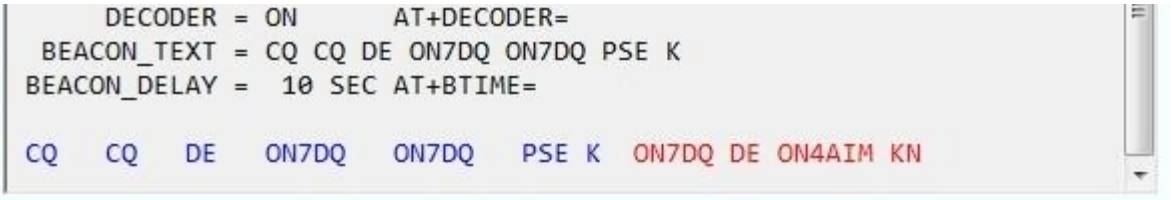
- One button to switch the **CW-Decoder** ON or OFF
- One button ot switch the **Generator** ON or OFF



There is a small change in the Memory Keyer.
Since the **F1** function key is reserved for the **HELP** function, the first 11 memories can be operated with the Function Keys **F2 to F12**. The further memories can only be controlled with the buttons on the screen (**M13 to M20**).

The Serial Monitor in the Windows program shows the decoded text in color:

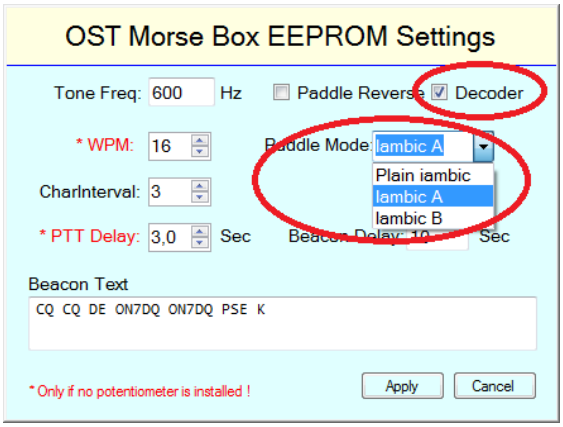
- **BLUE** = transmitted code from key or paddle
- **ROOD** = received code from pin D2
-



The **File Menu** has a new option: **Save Monitor Tekst**

The text in the monitor window can be saved as a **.txt** file (all black and white), or as a **.rtf** file with colored text.

The **File > EEPROM_Settings** menu now has an extra checkbox to enable or disable the decoder, and a dropdown list for the Iambic Mode.



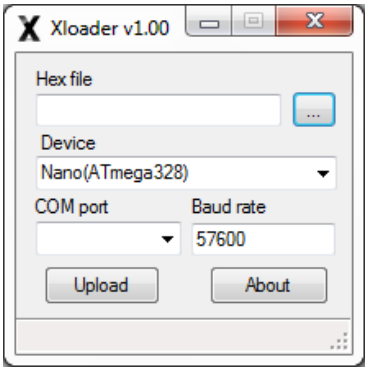
Programming the firmware with XLOADER

The program XLOADER provides an easy way to load the Arduino with new firmware, without need for the Arduino IDE or worries about special libraries.

Later updates can also be carried out just as easily. Download our github's Xloader zip file, and unpack everything in one folder. Make sure you have these files in that folder:

- devices.txt : it contains the needed device setting fort he Arduino Nano (and others)
- OST_Morse_V212.ino.eightanaloginputs.hex : this is 'the firmware'
  (the name can differ with newer versions)

Run XLoader.exe, and you should get this screen:



Select the hex file from the same folder as where you unzipped everything.

Check the device, it should say 'Nano(ATmega328)' (normally this should be OK from the start).
Find the COM port where the OST Morse Box is connected via a USB cable. The port number can be found in the Device Manager in Windows. Don't change the Baud Rate, it should be at 57600.

Finally click on 'Upload', and a few seconds later your Morse Box is ready to go!

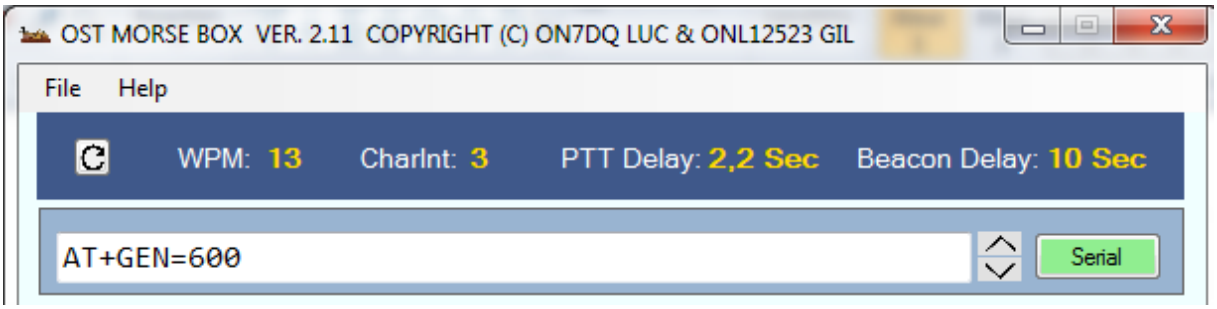
Adjusting the Decoder with the built-in Generator

Disconnect the transceiver from the Morse Box.

Connect the Morse Box to the PC, launch the Windows Program and make connection on the correct Serial Port (see basic manual).

Connect a clip lead from the right side of **R7** (where it is connected to trimmer R10) to the input of the decoder, this is best done on the top side of the potentiometer (or trimmer) **R4** in the Tone Decoder, or the potentiometer **R7** in the SMD Decoder.

Start the Generator with the AT command, this allows you to set the frequency per Hz.
For example, send **AT+GEN=600** to set the generator to exactly 600 Hz.



For the LM567 Tone Decoder, adjust the tone and level until the LED on pin 1 of the header lights up brightly.

For the SMD decoder you need to adjust the potentiometer for the level, and also the trimmer on the PCB .

Do a short test: loosen the clip lead and 'tap' some morse characters until they appear on the display.

References

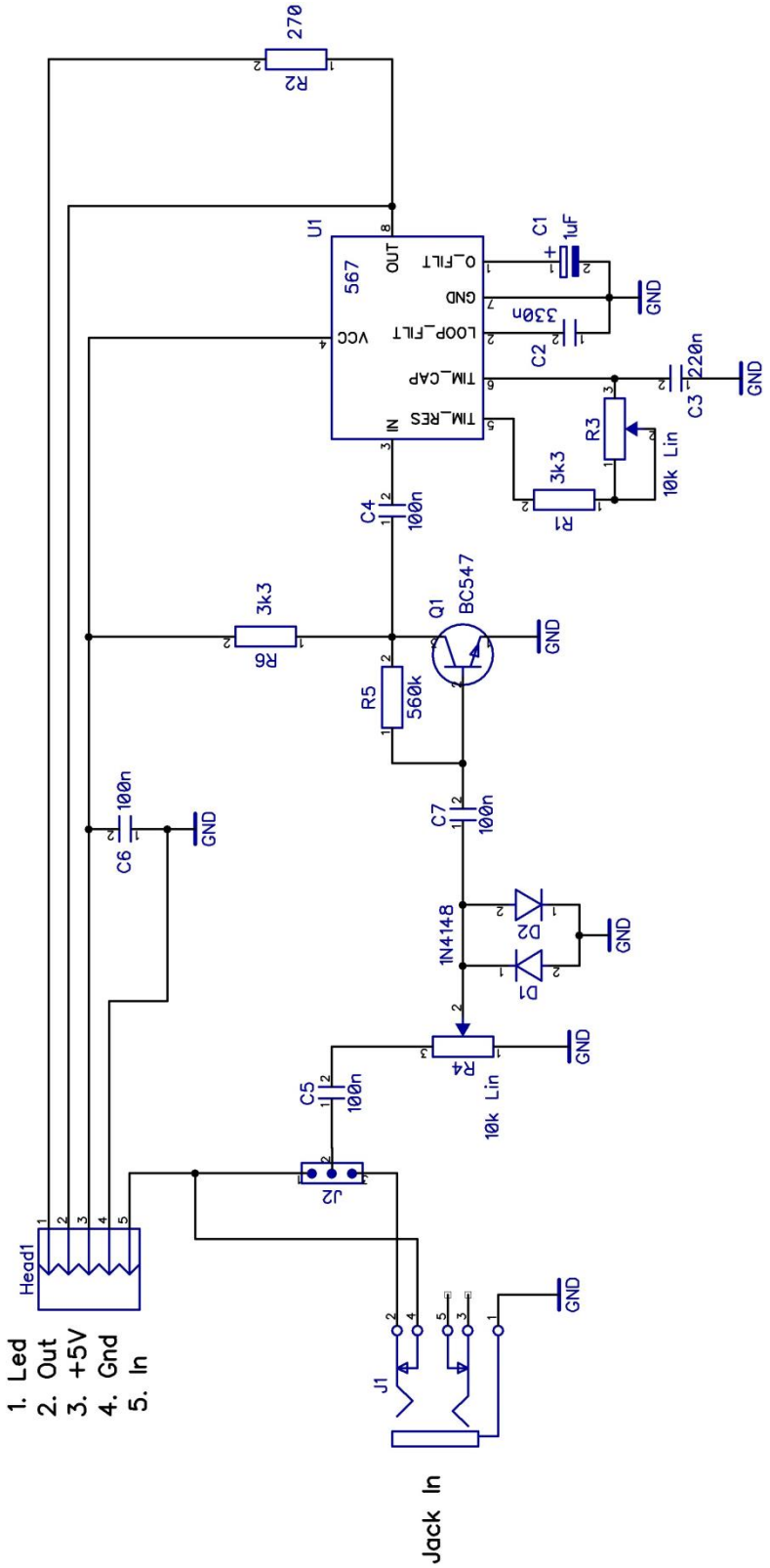
[1] Veron Morse Decoder: https://a08.veron.nl/zelfbouw/accessoires/cw-decoder/

[2] Elektor Labs: https://www.elektormagazine.nl/labs/morse-cw-audio-radio-telegraph-filter-1

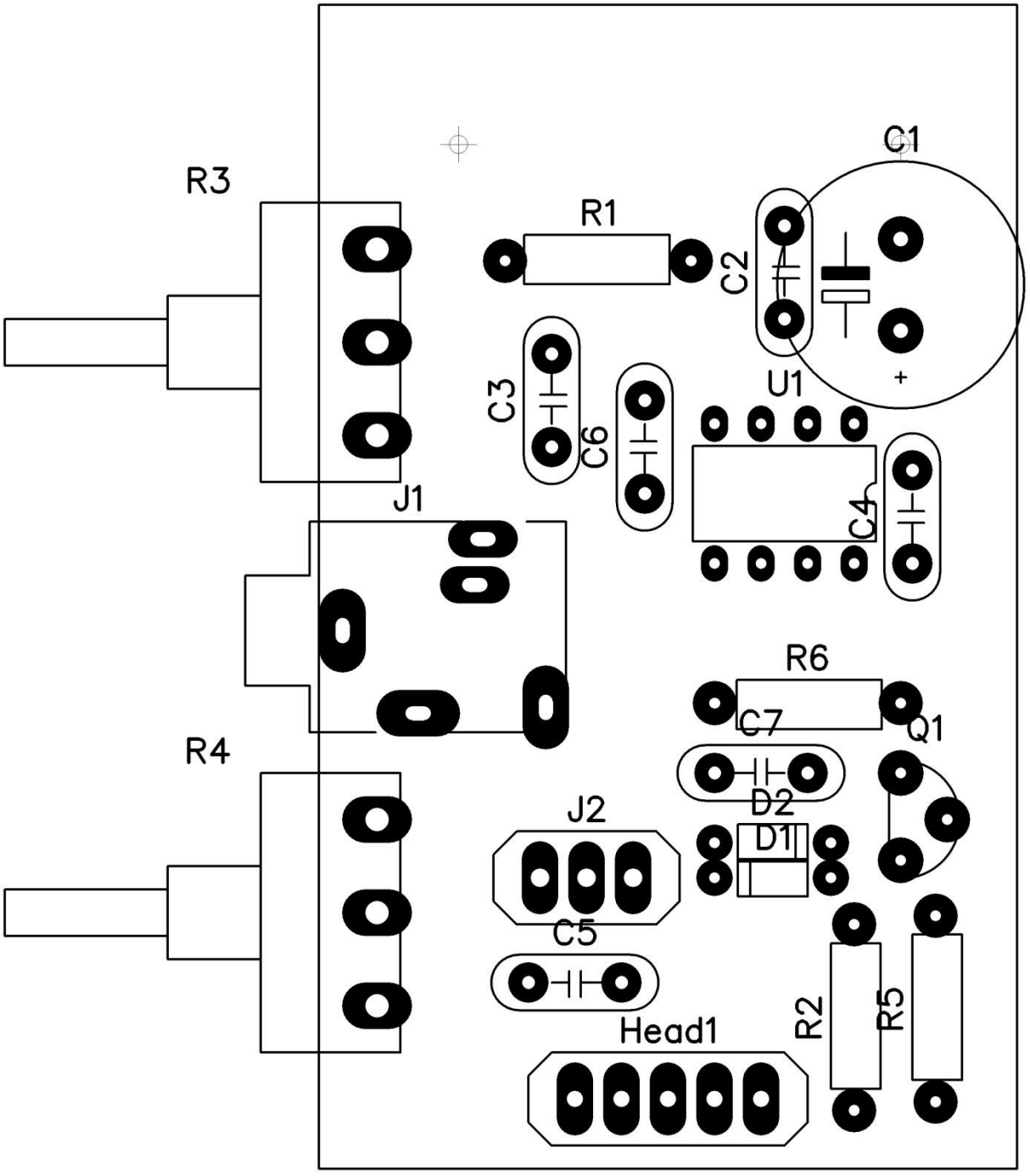
[3] Xloader master: https://github.com/binaryupdates/xLoader ,
ook te downloaden van http://www.hobbytronics.co.uk/download/XLoader.zip

Appendices

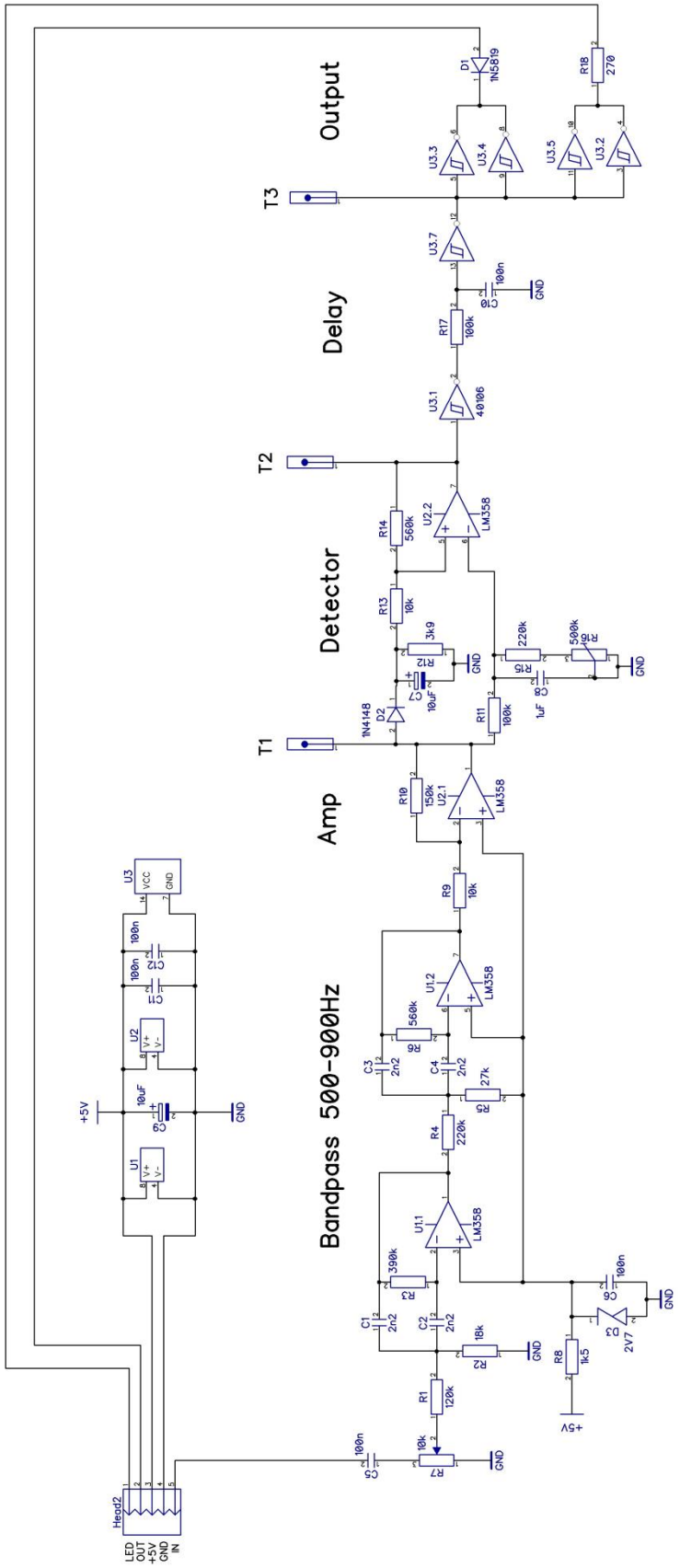
Circuit diagram - PLL Decoder



Parts placement - PLL Decoder



Circuit Diagram - BPF Decoder



Parts Placement - BPF Decoder