FbBalise – Beacon Controller

FbBalise is a beacon controller based on an ESP32-S3 microcontroller module.

It generates programmed sequences, scripted and stored in flash memory. A WEB interface is used to create or edit scripts and configure the system. The maximum size of a script is 4K characters.

Access is via a computer or smartphone by connecting to the « FbBalise » Wifi network, with the default password « 12345678 ». The access point name and password can be changed in the configuration.

FbBalise can also be connected to a local WiFi network.

Main features

- Browser-based configuration interface (Chrome, Opera, Firefox) over WiFi
- Power supply 7-18V (less than 40mA at 14V without WiFi)
- CW, OPERA, PI4, FT8, JT4, JT9, JT65, Q65, MSK144 and WSPR modes
- Time synchronisation and QTH locator calculation via GPS
- Can drive an AD9954 DDS
- Dimensions: (cm) 5.25 x 7.15

1. Getting started

FbBalise is configured through WEB pages accessible over Wifi. When powered on, a few seconds are needed for the Wifi to become active.

From your computer or smartphone, look for the « FbBalise » Wifi network and connect to it.

After connecting to the « FbBalise » Wifi, enter the password if prompted (« 12345678 » by default); a home screen will appear after a few seconds. This screen provides menu access to the application's various pages.

If you are connected to the « FbBalise » Wifi but the home screen does not appear, you can browse to http://192.168.4.1

FbBalise uses port 80 for both the http and the ws protocol.

If FbBalise is connected to a local network and you want public access, external port 80 must be opened. You can of course redirect a different external port to FbBalise's port 80.

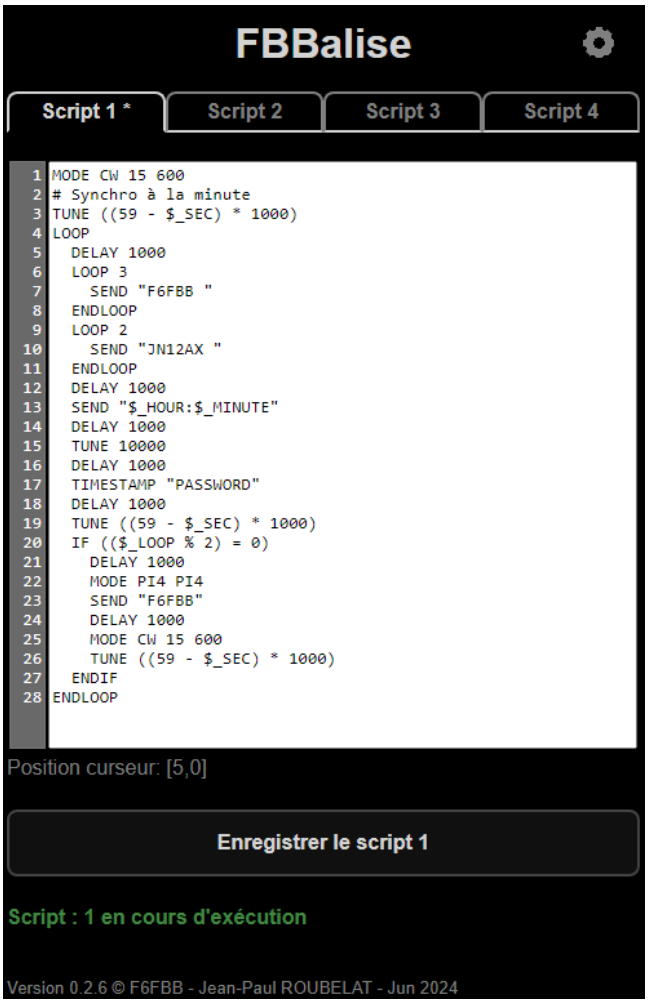
The tricolour LED indicates the operating state:

- 2 red flashes: initialisation phase and WiFi network search.
- Green flashes: board in WiFi Access Point mode
- Blue flashes: board connected to the WiFi network.

2. Home page

The home page gives access to:

- Configuration, via the gear icon in the top right of the window,
- Four tabs for managing and running four different scripts.
    - If a tab name is followed by an asterisk, it has been modified.
- An editable text window for creating or editing the script, with the cursor position shown below it.
- A line showing any error message or debug line.
- A button to start, restart or save a script depending on context
- A status line showing the script's execution state.



Illustrative screenshot. The version number shown at the bottom of the screen reflects the software version at the time of capture and may differ from the version described in this document (0.11.3).

3. Settings page

Pressing the gear icon in the top right takes you to the settings page.

This page allows you to change the default configuration and has four tabs:

- WiFi configuration,
- General settings,
- SPI
- Update.

> ⚠ **Warning**
> Any change must be saved using the « Validate » button

Three buttons at the bottom of the page let you choose a type of restart:

- « Restart »: FbBalise is restarted, settings are kept, no change is made to the PWS.
- « Erase flash »: the scripts stored in flash memory are erased. FbBalise restarts and the configuration is kept.
- « Factory settings »: all settings and scripts are erased, FbBalise restarts with the default configuration.

3.1. WiFi configuration

WiFi settings

- Name of the WiFi network (SSID) for the access point (AP).
- Password for access to this WiFi network. Default: « 12345678 ».

« WiFi delay » option

- This option is only active in access point (AP) mode. If FbBalise is connected to a WiFi network, it will not be disconnected.
- Wifi is activated when FbBalise is powered on.
- If the option is enabled, Wifi will automatically stop after the specified idle time in minutes, only in access point (AP) mode.
- The minimum delay is 5 minutes. Any page change or save action resets the delay to the specified value.
- If the option is enabled, the WiFi access point will only become available again after a power cycle, or by disabling/re-enabling it via « activation ».

Activation

- The WiFi access point can be disabled or enabled using a level of 0 (disabled) or 3.3V (enabled) on the specified spare input. If the spare input is not connected, the access point is enabled by default.

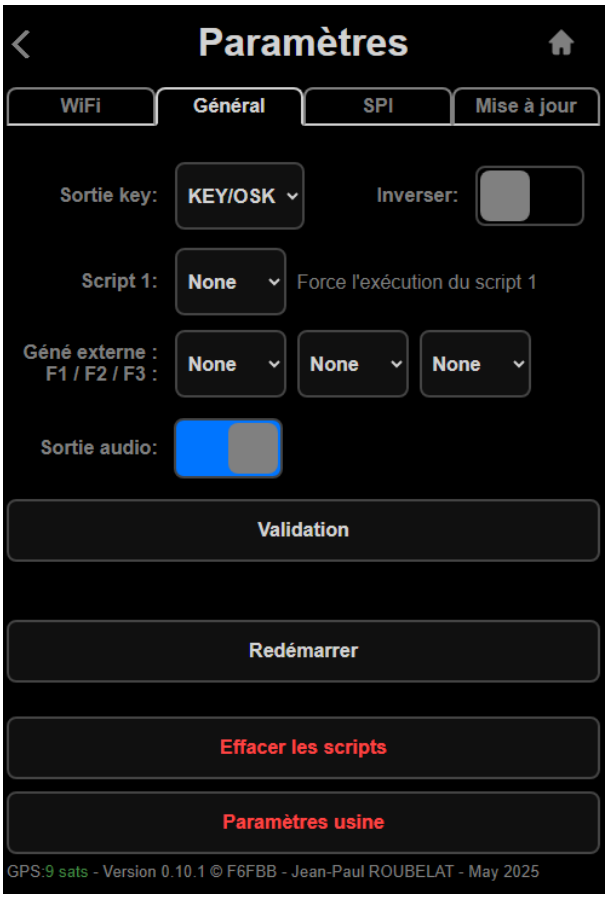
Internet WiFi access

- If a WiFi network and its password are specified, FbBalise will attempt to connect to that network. If it fails, it will start its own WiFi access point.

Configuration access via username / password

- If these fields are filled in, a username and password will be requested on connection.

3.2. General settings



Illustrative screenshot. The version number shown at the bottom of the screen reflects the software version at the time of capture and may differ from the version described in this document (0.11.3).

CW KEY signal output

- Selects the CW Key output port (spare 1 to 6)

CW KEY signal inversion

- Inverts the CW output signal

Script 1

- Lets you select a spare that will lock the beacon onto script 1.

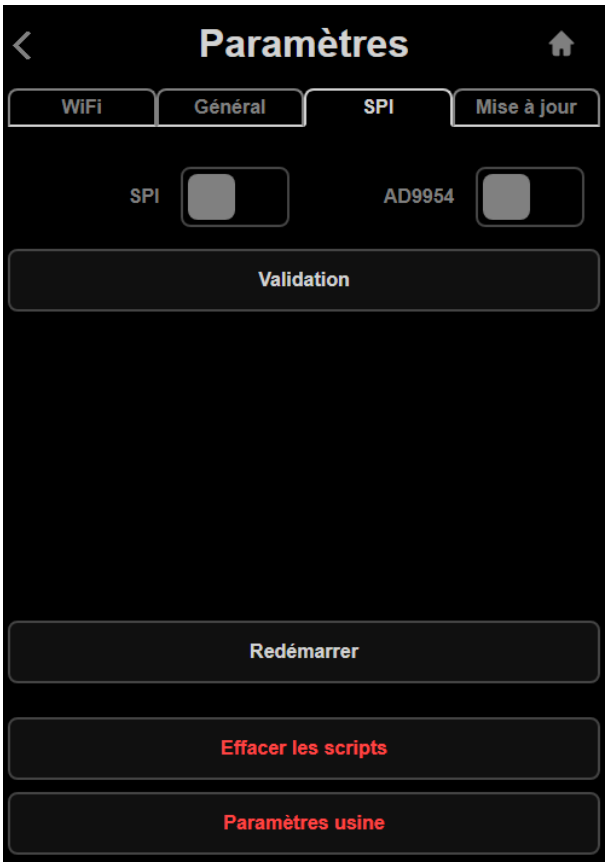
External generator

- Specifies three spares that provide the phase used to configure an external generator. See §9 External generator.

Audio output

- Enables or disables audio signal generation. A restart is required for the change to take effect.

3.3. SPI configuration



If both the SPI and AD9954 switches are enabled, the SPI configuration page allows an AD9954-type DDS to be programmed. See §12.

3.4. Update

Illustrative screenshot. The version number shown at the bottom of the screen reflects the software version at the time of capture and may differ from the version described in this document (0.11.3).

This page is used to update the firmware with a new software version. The file is of the form « file_name.bin ».

After entering the file name, press the « Update » button. The file will be sent to the beacon module for updating. Once the update is complete, the module will restart.

> ⚠ **Warning**
> Do not unplug the module during the update phase.

It is also possible to save and restore the configuration to/from a JSON file.

4. Script keywords

- A parameter in square brackets is optional
- An expression can be
    - a numeric value
    - a string of characters in quotes
    - an expression in parentheses, containing expressions, variables, numeric values, parentheses and the signs +, -, /, *, %
    - a condition in parentheses, containing operations, variables, numeric values, parentheses and the signs >, >=, <, <=, =

4.1. DEBUG / ECHO

- DEBUG takes a text parameter displayed on the home page, below the edit window.
- ECHO is identical to the DEBUG keyword but does not display the DEBUG + line number information.
- Use quotes to define the argument, as it may contain spaces.
- If the argument contains a variable, its content will be converted and displayed as text.
- A variable can be formatted, see §5.3 Formatting a variable.

In the following example, an external voltage divider with a ratio of 5.12 is assumed. The variable will always be displayed with one decimal place.

```
DEBUG "VOLTAGE: $($_A2 * 5.12):1 VOLTS UPTIME : $_UPTIME"
```

4.2. DELAY

- DELAY takes an expression parameter specifying the pause delay in milliseconds

```
DELAY (EXPRESSION)
```

4.3. HEX

> ⚠ **Warning**
> The HEX function described below is not yet implemented in version 0.11.3. The behaviour and syntax shown are those planned for a future version.

- Function that converts a number or expression into a hexadecimal string. It expects two parameters, given as expressions:
    - A number, of which only the integer part is used
    - The number of bits of the hexadecimal number. If the number is larger than this length, it will be truncated to this length.

```
HEX((4.52*10), 12) RETURNS THE STRING « 02D »
```

- This function does not preserve the sign; that is the integrator's responsibility. So, at 12 bits, 4095 and -1 will both return « FFF »

4.4. IF / ELSIF / ELSE / ENDIF

- IF or ELSIF take a parameter in parentheses defining the condition.
    - If the condition is true, only the IF/ELSIF branch is executed. Only one IF/ELSIF branch will be executed; if no IF or ELSIF is true, the ELSE branch is executed.
    - There can be several ELSIF blocks.
    - If the condition equals 0, only the ELSE branch is executed.
- An ENDIF must always mark the end of the conditional test.

```
IF (CONDITION)
```

```
...
ELSIF (CONDITION)
...
ELSE
...
ENDIF
```

4.5. LOOP / ENDLOOP

- LOOP can take an optional parameter specifying the number of iterations
- If the number of iterations is not specified, the loop is infinite
- An ENDLOOP must always mark the end of the loop.

```
LOOP [(EXPRESSION)]
...
ENDLOOP
```

4.6. MODE

- MODE takes one or more parameters specifying the mode options
    - CW (speed in words per minute) [tone] — default tone 800Hz
    - OPERA PERIOD [tone] — PERIOD one of OPERA05, OPERA1, OPERA2, OPERA4, OPERA8, OPERA32, OPERA65, OPERA2H; default tone 1500Hz
    - PI4 Type [tone] — TYPE one of PI4, PI4-80, PI4-96, PI4-120; default base tone 800Hz
    - JT Type [tone] — TYPE one of FT8, JT4A→JT4G, JT65A→JT65C, JT9A→JT9H, Q65-15A→Q65-15C, Q65-30A→Q65-30D, Q65-60A→Q65-60E, Q65-120A→Q65-120E, Q65-300A→Q65-300E, WSPR; default base tone 800Hz for JT/Q65xx and 1500Hz for WSPR
    - JT MSK144 [nb_of_frames] — no tone configuration; centre frequency is 1500Hz with a +/-500Hz offset. A frame lasts 72ms; a 30-second transmission requires about 400 frames.

```
MODE CW (EXPRESSION) [(EXPRESSION)]
MODE OPERA OPERA05 [(EXPRESSION)]
MODE PI4 PI4 [(EXPRESSION)]
MODE JT JT4G [(EXPRESSION)]
```

4.7. RESTART

- Does not require a parameter,
- Restarts FbBalise. The configuration and scripts are kept.

```
IF ($_UPTIME >= 24) # Restarts FbBalise every 24 hours
  RESTART
ENDIF
```

4.8. SEND

- SEND sends a text passed as an argument, using the mode previously defined by the MODE keyword.
- Use quotes to define the argument, as it may contain spaces.
- If the argument contains a variable, its content will be converted to text
- A variable can be formatted, see §5.3 Formatting a variable.
- The content may be constrained by the mode in use.
- For OPERA, only a valid callsign will be accepted as an argument.
- For JT, the callsign and QTH locator can be sent, as well as WSPR message types 1, 2 or 3.

```
SEND "F6FBB JN12BT "
SEND "VOLTAGE $($_A2 * 4.1):1 " # 1 decimal place
```

4.9. SET

- SET defines or redefines the value of a variable.
- A variable name must start with the $ character.
- A variable is always numeric.
- A variable can be used in an expression.
- SET takes two parameters:
    - A parameter specifying the variable name. This name must start with a consonant and be strictly alphanumeric.
    - An expression parameter specifying the numeric value assigned to the variable

```
SET $MYVAR ($OTHERVAR / 12.5)
SET $MYVAR 12.5
SET $MYVAR ($MYVAR + 1) # increments $MYVAR
```

4.10. TUNE

- TUNE takes an expression parameter specifying the carrier duration in milliseconds

```
TUNE (EXPRESSION)
```

4.11. TIMESTAMP

- TIMESTAMP takes a text parameter specifying the password used to compute the key. This password MUST be in uppercase. It follows the G4JNT specification.
- The current mode must be CW.
- This command only works if the GPS is connected.
- The time used for the calculation is the GMT time provided by the GPS.
- If the time is not valid (i.e. no GPS), the message « TIME ? » will be sent.

```
TIMESTAMP PASSWORD
```

> The result of the calculation is also available read-only via the predefined variable $_TIMEST[PASSWORD] (see §5.2), which can be used in a SEND or DEBUG statement outside of the TIMESTAMP keyword.

5. Variables

5.1. Declaring and using variables

Variables are always numeric. They are defined or redefined using the « SET » keyword. A special variable $(EXPRESSION) needs no declaration; it contains the expression to be evaluated and represents its result.

If a variable is used on a « SEND » line, its value will be formatted and converted to text as needed. The number of variables is limited to 10.

> A variable name can be abbreviated, provided the abbreviation remains unambiguous with respect to the other variables used in the script. For example, $_SEC can be used instead of $_SECOND, or $_HUMI instead of $_HUMID, as long as no other variable starts with the same letters. This makes it possible to write more compact scripts, which is useful given the 4K-character limit per script.

5.2. Predefined variables

Several system-related variables are predefined. They are always numeric variables. Their name always starts with an underscore, except for $(expression). Some are read-only (no initialisation needed), others are write-only.

Variable	Value	Comment	R/W
`$_I1 to $_I6`	0 or 1	Spare input state 0=LOW 1=HIGH	R
`$_O1 to $_O6`	0 or <> 0	Sets the spare 0=LOW otherwise HIGH	W
`$_A1 to $_A3`	0 to 3.3	Analog input voltage in volts	R
`$_T0 to $_T9`	Float or Text	Telemetry for the Windows application	W
`$_PTT`	0 or 1 (PTT on)	Sets the PTT output	W
`$_HOUR`	0 to 23 or -1	Returns -1 if GPS is not active	R
`$_MINUTE`	0 to 59 or -1	Returns -1 if GPS is not active	R
`$_SECOND`	0.000 to 59.999 or -1	Returns -1 if GPS is not active. With ms	R
`$_DAY`	1 to 31 or -1	Returns -1 if GPS is not active	R
`$_MONTH`	1 to 12 or -1	Returns -1 if GPS is not active	R
`$_YEAR`	yyyy or -1	Returns -1 if GPS is not active	R
`$_UPTIME`	Number of hours	Number of hours since startup (float)	R
`$_UPSCRIPT`	Number of hours	Number of hours since the script started	R
`$_LOOP`	Number of iterations	Number of iterations in the active loop	R
`$_POWER`	Voltage	Gives the module's supply voltage	R
`$_TEMPE`	Temperature	Or -1 if BME280 not connected	R
`$_HUMID`	Humidity	Or -1 if BME280 not connected	R
`$_BAROM`	Barom. pressure	Or -1 if BME280 not connected	R
`$_ALTITUDE`	Altitude (metres)	Or -1 if BME280 not connected	R
`$_BALTI`	Altitude correction (m)	Barometric pressure correction.	RW
`$_CALLSIGN`	Callsign	Text only, as defined in the settings	R
`$_LOCATOR`	QTH Locator	Text only, computed from the GPS	R
`$_TIMEST[.]`	Time stamp	Text only. Corresponds to the calculation performed by the TIMESTAMP keyword (see §4.11). Password in brackets.	R

> The variable $_TIMEST[PASSWORD] contains the timestamp in the format specified by G4JNT, i.e. the minute as 2 digits, a space, and 3 letters. The password in brackets MUST be in uppercase.

> The $_BALTI variable defaults to 0, i.e. no pressure correction. Its value must be initialised with the altitude for the correction to apply.

> ⚠ **Warning**
>
> The $_TEMPE, $_HUMID and $_BAROM variables require a BME280 connected to the I2C port.

Address detection is automatic.

> ⚠ **Warning**
> Variables $_I1 to $_I6 and $_A1 to $_A3 configure the spare pin as an input. Variables $_O1 to $_O6 configure the spare pin as an output. If the spare pin in question is used for another purpose, this may affect its operation.

> ⚠ **Warning**
> The input voltage on a spare pin must not exceed 3.3V, or it may destroy the microcontroller. Higher voltages can be measured using an external voltage divider. The value can then be multiplied using an expression.

Examples:

```
SET $_O2 1   # Sets spare 2 to high level
SET $_O2 0   # Sets spare 2 to low level
IF ($_I1)    # True if spare 1 is at high level
IF ($_A3 > 2.1)   # True if analog port 3 voltage is above 2.1V
SET $_T0 "BATTERY VOLTAGE: $_POWER:2,V"   # Sends the supply voltage to the Windows
application on telemetry channel 0
```

5.3. Formatting a variable

When a variable is used in text form (in DEBUG or SEND for example), it can be formatted using the following syntax:

```
$VAR:N,S
```

- N is the number of decimal places. The value will be rounded accordingly.
- S is the separator character that will replace the decimal point.
- Formatting is optional but, when used, must follow this order

Examples:

```
SET $PI 3.1415926536   # Initialises the variable with the value 3.1415926536
SEND "$PI:3,V"   # Sends the text 3V142
SEND "$PI:,V"    # Sends the text 3V14159
```

Supply voltage and weather variables for PI4:

```
SEND "/_PS$_POWER:1,V"   # Sends the voltage 12.4v: '/_PS12V4'
SEND "/_HUMI$_HUMI:0"    # Sends the humidity 56%: '/_HUMI56'
IF ($_TEMPE >= 0)   # Case for +26° and -12°
  SEND "/_TMPO$_TEMPE:0"   # Sends the temperature 26°: '/_TMPO26'
ELSE
  SEND "/_TMNE$(-$_TEMPE):0"   # Sends the temperature -12°: '/_TMNE12'
ENDIF
SET $_BALTI 400   # Barometric pressure correction using altitude (400m)
SEND "/_BARO$($_BAROM%100):0"   # Sends the pressure 1016hPa: '/_BARO16'
```

6. Expressions

An expression has the form « A operator B » where:

- A is a value, a variable or an expression.
- An operator from the table below,
- B is a value, a variable or an expression.

Operations are performed in floating point. The operators are:

Operator	Action	Comment
+	Addition	
-	Subtraction	
*	Multiplication	
/	Division	Quotient of the floating-point division
%	Modulo	Remainder of the floating-point division

Multiplications, divisions and modulo are performed first, left to right. Additions and subtractions are performed next, also left to right.

An expression can be enclosed in parentheses, in which case the parenthesised expressions are evaluated first.

Examples:

```
(3 + 5 * 2)   => 3 + 10 => 13
((3 + 5) * 2) => 8 * 2  => 16
(3 * 5 + 2)   => 15 + 2 => 17
```

7. Comparisons and logical operators

A comparison has the form « A Comparator B » where:

- A is a value, a variable or an expression.
- B is a value, a variable or an expression.

Calculations are done in floating point, so equality may be hard to achieve exactly. The result of a comparison is 1.0 if true, otherwise 0.0 if false.

Logical operators also work on values equal to 0 (logical 0) or different from 0 (logical 1). The operators are:

Comparator	Action	Comment
<	Less than	
>	Greater than	
<=	Less than or equal	
>=	Greater than or equal	
!= or <>	Not equal	
== or =	Equal	
&	Logical AND	
|	Logical OR	
^	Logical XOR	

Examples:

```
IF ($_A3 > 2.1)   # True if analog port 3 voltage is above 2.1V
  Executed if analog3 is above 2.1V
ELSE
  Executed if analog port 3 voltage is at or below 2.1V
ENDIF
IF ($_T & 2.1)   # True if the value of $_T is different from 0
```

8. Writing a script

8.1. Creating a script

The script is typed (or pasted) into the edit window. Once editing is finished, the save button sends and saves the script to the ESP32's flash memory.

- The script is interpreted in uppercase and is not case-sensitive.
- Every line must start with a keyword, followed by its arguments if any.
- Any blank line is ignored.
- Everything after the # character on a line is ignored

8.2. Simple script

This script will:

- Select CW mode at 15 words per minute, tone 800Hz
- Send the callsign and QTH locator followed by a 1s pause, a 5s tune, a 1s pause
- All in an infinite loop.

```
MODE CW 15 800
LOOP
  SEND "F6FBB JN12BT "
  DELAY 1000
  TUNE 5000
  DELAY 1000
ENDLOOP
```

8.3. More advanced script

This script will generate:

- A CW mode at 15 words per minute, 800Hz,
- The callsign three times, the QTH locator, and a 500ms pause,
- Then a 500ms pause, a 5-second tune, then a 1-second pause
- It will loop indefinitely…

```
MODE CW 15 800
LOOP
  LOOP 3
    SEND "F6FBB JN12BT "
    DELAY 500
  ENDLOOP
  DELAY 500
  TUNE 5000
  DELAY 1000
ENDLOOP
```

8.4. Script with a conditional test

This script will generate:

- A CW mode at 15 words per minute, 800Hz,
- The callsign three times, a 1s pause, then the QTH locator twice,
- Then a 500ms pause, a 5-second tune, then a 1-second pause
- Every 4 loops it will send the callsign in OPERA mode then return to CW mode
- It will loop indefinitely…

```
MODE CW 15 800
LOOP
  LOOP 3
    SEND "F6FBB "
  ENDLOOP
  DELAY 1000
  LOOP 2
    SEND "JN12BT "
  ENDLOOP
  DELAY 1000
  TUNE 5000
  IF (($_LOOP % 4) = 0)
    DELAY 1000
    MODE OPERA OPERA05 1500
    SEND "F6FBB"
    MODE CW 15 800
  ENDIF
  DELAY 1000
ENDLOOP
```

8.5. Script with minute synchronisation

This script sends the callsign 3 times followed by a 1-second pause, then tunes until second 59, pauses for 1 second, then sends the callsign in OPERA mode. It requires the GPS module to be present in order to have the time information.

```
MODE CW 15 800
LOOP
  LOOP 3
    SEND "F6FBB "
  ENDLOOP
  DELAY 1000
  TUNE ((59 - $_SEC) * 1000)
  DELAY 1000
  MODE OPERA OPERA05 1500
  SEND "F6FBB"
  DELAY 1000
  MODE CW 15 800
ENDLOOP
```

9. External generator

In the « Frequencies » tab of the configuration menu, it is possible to select 3 spares to drive an external OOK or frequency generator.

The levels of the spares assigned to F1, F2 and F3 follow this logic:

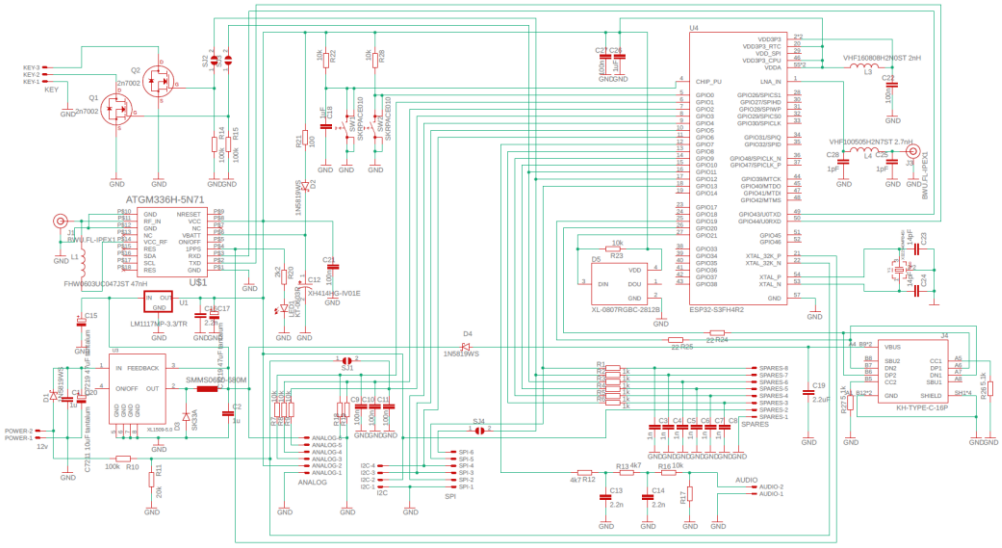
F3	F2	F1	Frequency to generate
0	0	0	CW_OFF / SPACE — OOK or shift generation depending on hardware
0	0	1	CW_ON / MARK — OOK or shift generation depending on hardware
0	1	0	OPERA_OFF / SPACE — OOK or shift generation depending on hardware
0	1	1	OPERA_ON / MARK — OOK or shift generation depending on hardware
1	0	0	PI4/0 — PI4 frequency 0
1	0	1	PI4/1 — PI4 frequency 1
1	1	0	PI4/2 — PI4 frequency 2
1	1	1	PI4/3 — PI4 frequency 3

- Selecting all 3 spares is not mandatory. It is possible to select any subset of them.
    - F1 => « KeyUp / KeyDown » in the case of a CW-only beacon
    - F2 => CW / OPERA if PI4 is not used
    - F3 => PI4 mode

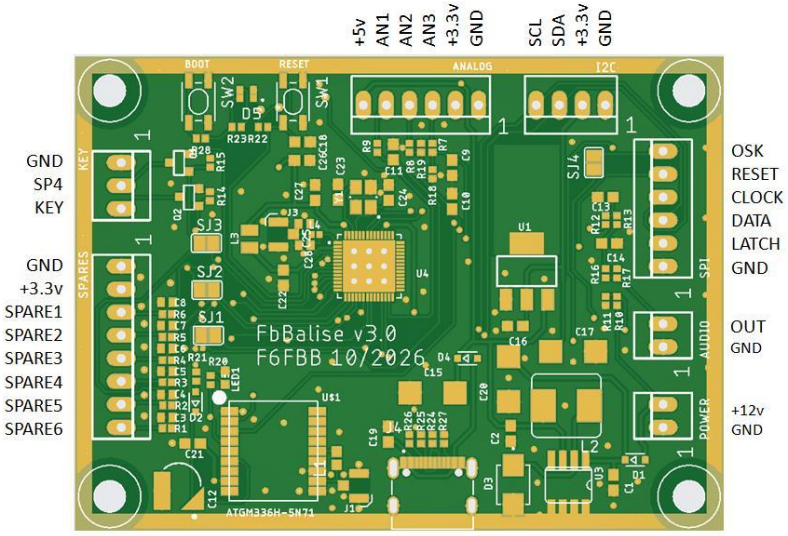
Example: this script activates the external generator in CW mode (F2=0), sends the callsign, then toggles F1 (KeyUp/KeyDown) according to the text being sent, before returning the spares to a low level:

```
SET $_O2 0   # F2 = 0: CW mode
MODE CW 15 800
LOOP
  SET $_O1 1   # F1 = 1: CW_ON / MARK
  DELAY 500
  SET $_O1 0   # F1 = 0: CW_OFF / SPACE
  DELAY 500
ENDLOOP
```

10. Circuit diagram



11. Connectors and jumpers



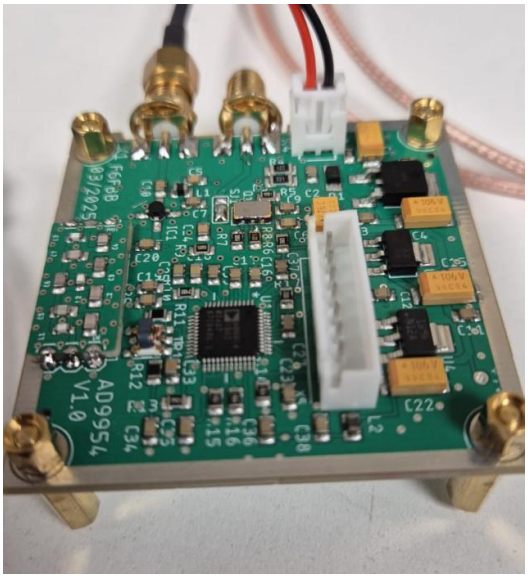
- SJ1: supply voltage measurement on AN1
- SJ2: open-drain OOK output
- SJ3: open-drain PTT output
- SJ4: SPI OOK output

- J1: GPS antenna
- J3: WiFi antenna

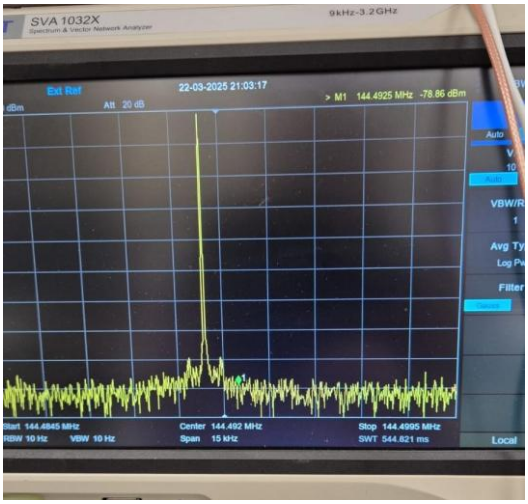
12. AD9954 DDS interface and board

A board based on Analog Devices' AD9954 DDS chip can be driven directly over the SPI bus.

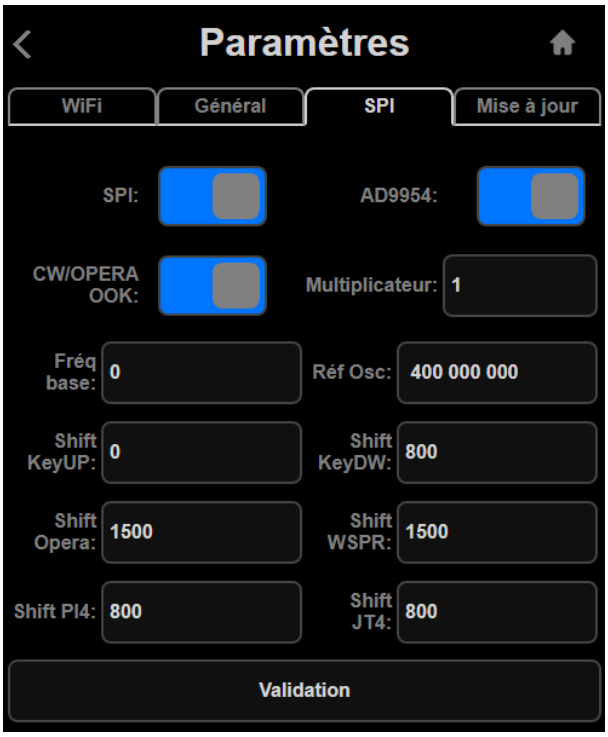
If the SPI switch and the AD9954 switch are enabled, a dedicated window allows the chip to be configured and the RF signal to be generated in FSK or OOK, depending on the modes and settings.



AD9954 DDS daughter board



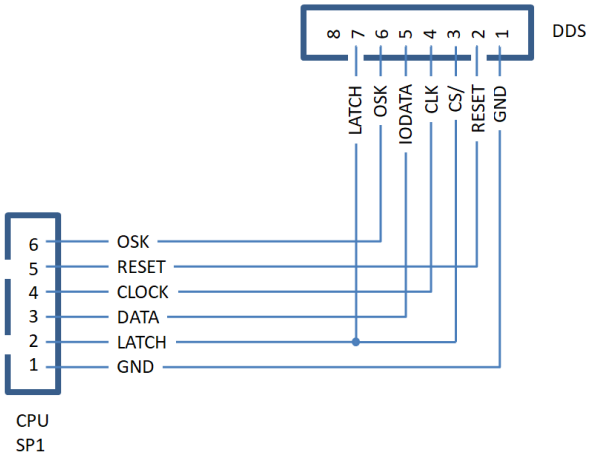
Measurement on a spectrum analyser



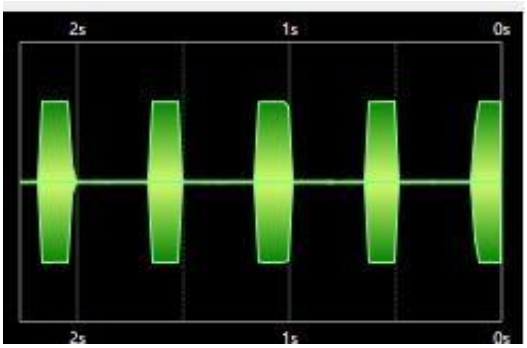
The base frequency value must be entered. The maximum frequency is 160,000,000 Hz.

All shift and frequency values are expressed in Hz.

The multiplier is used to adapt the shift values of the multi-frequency modulation modes to a multiplication of the generated frequency. For example, in the case of a 144MHz signal multiplied by 3 to generate 432MHz, the 315Hz shift in JT4G would be reduced to 105Hz, so as to recover the nominal value after the transmitter's multiplication stages. The other modes, PI4 and WSPR, work the same way.



OOK modulation generation with anti-key-click edges



FbBalise-to-DDS-connector cable

13. Keyword quick reference

This table summarises, for quick reference, all the script-language keywords described in §4.

Keyword	Role
`DEBUG / ECHO`	Displays a text (with variables) on the home page
`DELAY (EXPRESSION)`	Pause in milliseconds
`HEX(n, bits)`	Converts a number to a hexadecimal string (not implemented in 0.11.3)
`IF / ELSIF / ELSE / ENDIF`	Conditional test
`LOOP [(n)] / ENDLOOP`	Loop, infinite if n is absent
`MODE ...`	Selects the transmission mode (CW, OPERA, PI4, JT, MSK144…)
`RESTART`	Restarts FbBalise (config. and scripts kept)
`SEND "text"`	Sends a text in the current mode
`SET $VAR (EXPRESSION)`	Defines or changes a variable
`TUNE (EXPRESSION)`	Sends a carrier for a given time (ms)
`TIMESTAMP PASSWORD`	Sends an encrypted timestamp (G4JNT standard), requires GPS and CW mode

14. Acknowledgements

The software was designed and written by me. It runs on an FbBalise board and can use a GPS daughter board (for time synchronisation and location).

The software is written in C/C++ using the « platformio / vscode » environment and includes several third-party ESP32 libraries.

Thanks to F5DJL and F1HDI for their ideas, feedback and support!

Jean-Paul ROUBELAT, F6FBB.