FbBalise – Beacon Controller

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



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

Access is from a computer or smartphone by connecting to the “FbBalise” WiFi network, with the default password “12345678”. The access point name and its password can be changed in the configuration.

FbBalise can also be connected to a local WiFi network.

Main features:

- Configuration interface in a web browser (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, WSPR and JTTY modes
- Time synchronisation and QTH locator computed from GPS
- Can drive an AD9954 DDS
- Dimensions: (cm) 5.25 x 7.15

Table of contents

1. What's new in version 0.16.x 4
2. What's new in version 0.15.0 5
3. Getting started 6
4. Home page 7
5. Settings page 8
   5.1. WiFi configuration 9
      WiFi settings 9
      "WiFi delay" option 9
      Activation 10
      Access through an Internet WiFi network 10
      Configuration access with user name / password 10
   5.2. General settings 11
      Callsign 11
      Language 11
      CW KEY signal output (active if SPI is OFF) 11
      CW KEY signal inversion 12
      1PPS output (active if SPI is OFF) 12
      Script 1 12
      External generator 12
      Audio output 12
   5.3. SPI configuration 13
   5.4. MQTT 14
   5.5. Infos 15
   5.6. Update 16
   5.7. Logout 17
6. Script keywords 18
   6.1. BREAK 18
   6.2. DEBUG / ECHO 18
   6.3. DELAY 18
   6.4. HEX (to be implemented) 19
   6.5. IF / ELSIF / ELSE / ENDIF 20
   6.6. LOOP / ENDLOOP 20
   6.7. MODE 21
   6.8. RESTART 21

6.9. SEND....23
6.10. SET....23
6.11. TUNE....23
6.12. TIMESTAMP....24
7. Variables....25
7.1. Declaring and using variables....25
7.2. Predefined variables....25
7.3. Telemetry variables....26
7.4. Formatting a variable....27
8. Expressions....29
8.1. Functions....29
9. Comparisons and logical operators....31
10. Writing a script....32
10.1. Creating the script....32
10.2. Checking and autonomous operation....32
10.3. Simple script....32
10.4. More advanced script....33
10.5. Script with a conditional test....33
10.6. Script synchronised on the minute....34
10.7. Synchronised WSPR script....34
11. External generator....35
12. Schematic....36
13. Connectors and jumpers....37
14. AD9954 DDS interface and board....38
15. Acknowledgements....40

1. What's new in version 0.16.x

New mode

- New JTTY digital mode, from WSJT-X 3.2.0-rc1: MODE JT JTTY [tone]. Unlike the other JT modes (callsign/locator or standard messages), JTTY sends free text, which can be used for example for telemetry (see the SEND keyword).
- Frame of 59 symbols (13 for synchronisation + 46 for data) at 31.25 baud, i.e. about 1.9s per frame of 5 characters; up to 16 frames (80 characters) per SEND.
- Base tone 1500Hz by default, as for WSPR.

This mode is limited to the AD9954 DDS for now (the LMX2571 is not yet supported for this mode).

GesBal telemetry

- The beacon receives over the local network the measurements of the GesBal board associated with it: voltage, current, forward and reflected power, SWR and temperatures. No setting is needed on the beacon: the association is made in GesBal (version 0.2.6 or later) with the MAC address of the beacon.
- New predefined variables $_GV, $_GI, $_GP, $_GR, $_GS, $_GT, $_GU (associated board), $_GV1 to $_GU3 (explicit board), $_GBN (associated board) and $_GAG (age of the data), usable in SEND, SET, IF and the telemetry: see "GesBal telemetry variables".
- Infos tab: new "GesBal:" line with the associated board, its measurements and the age of the data, shown as soon as the first data is received.
- Reception does not interfere with transmission: it is handled as a low-priority background task, with no memory allocation and no flash write.

Software update

- Each .bin file now contains the project name and its version. A binary from another project, or built for another chip, is rejected, whether the update is done from the Update page, by MQTT or over the network (OTA).
- The Update page shows the identity of the selected file (name, version, build date) before sending it. An old binary without identification is only accepted if "Force" is ticked.
- A rejected update shows the reason for the rejection, and the beacon keeps running without restarting.
- MQTT: the "update force" command allows going back to an old version without identification.

2. What's new in version 0.15.0

General

- The beacon frequency in the SPI tab is now the beacon transmit frequency, after any external multiplication.

Script editor

- Automatic indentation: two spaces per LOOP and IF level on display, cursor placed at the right level after Enter, Tab key to re-indent everything. Leading spaces are no longer saved.
- Block checking while typing (unclosed LOOP/IF, misplaced end of block or BREAK), shown in orange.

Scripting language

- DELAY and TUNE in absolute time: UNTIL s [EVERY p], in UTC seconds (example: DELAY UNTIL 1 EVERY 120 for an even minute).
- New BREAK keyword to exit a loop.
- Functions in expressions: RND, INT, ROUND, ABS, MIN, MAX.
- New predefined variables $_TOD (seconds since midnight UTC) and $_DOW (day of the week).
- The LOOP count can be a variable or an expression; LOOP 0 does not run the loop.
- The # character is accepted inside a quoted text.

Reliability of autonomous operation

- The whole script is checked when it is saved and when it is started; a script with an error does not start.
- The last valid version of each script is kept and run instead of a version saved with an error.
- Once started, the script no longer stops on an error: the line concerned is skipped and reported only once.
- $_SCRIPT only switches script if the requested script is usable.
- Variables are reset each time a script is started.
- DELAY and TUNE are timed to the microsecond, with no practical duration limit.
- The configuration is only rewritten to flash memory if it has changed.

Fixes

- ELSIF was executed even when a previous branch had already been executed.
- After 10 variables had been created, no variable could be modified any more.
- Two consecutive variables in a text (AB) were not both replaced, and a long text containing variables could overflow.
- An unknown mode (MODE XYZ) was accepted without error.
- TIMESTAMP reported a mode error even in an IF branch that was not executed.

3. Getting started

FbBalise is configured through web pages accessed over WiFi. At power-up, a few seconds are needed for the WiFi to be 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 needed (“12345678” by default); a welcome window appears after a few seconds. From this window, a menu gives access to the different pages of the application.

If you are connected to the “FbBalise” WiFi but the window does not appear, you can open it at the address http://192.168.4.1

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

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

The three-colour LED shows the operating state:

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

Two buttons:

- SW1: reset.
- SW2: boot. A long press while running (more than 10s) clears the board login and password, in case the password has been forgotten. When it is taken into account, the LED flashes quickly in white.

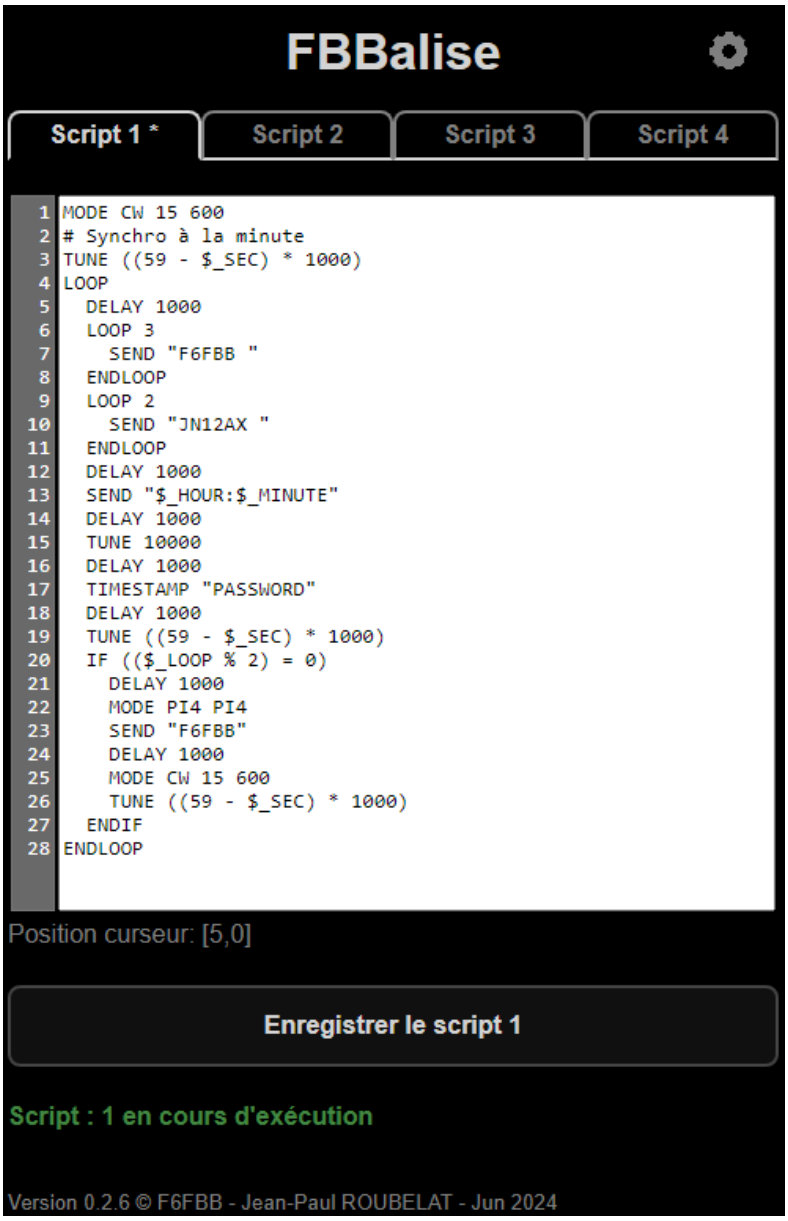
Two IPX/U.FL antenna sockets:

- J1: GPS.
- J3: WiFi.

4. Home page

The home page gives access to:

- The configuration, through the gear wheel at the top right of the window,
- Four tabs to manage and run four different scripts.
    - If a tab name is followed by an asterisk, the script has been modified.
- An editable text window to create or modify the script, with the cursor position below the window.
- A line showing any error message or a debug line.
- A button to start, restart or save a script, depending on the context
- A status line giving the running state of the script.



5. Settings page

Pressing the gear wheel at the top right opens the settings page.

This page is used to change the default configuration and has four tabs:

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



Any change must be saved with the “Validation” button

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

- “Restart”: FbBalise is restarted, the settings are kept, nothing is changed.
- “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.

5.1. WiFi configuration



WiFi settings

- WiFi network name (SSID) for the access point (AP).
- Password to access 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.
- The WiFi is switched on when FbBalise is powered up.
- If the option is enabled, the WiFi switches off automatically after the specified idle time in minutes, only in access point (AP) mode.
- The minimum delay is 5 minutes. Any page change or save restarts the specified delay.
- If the option is enabled, the WiFi access point will only be available again after a power cycle, or by disabling/enabling it with “Activation”.

Activation

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

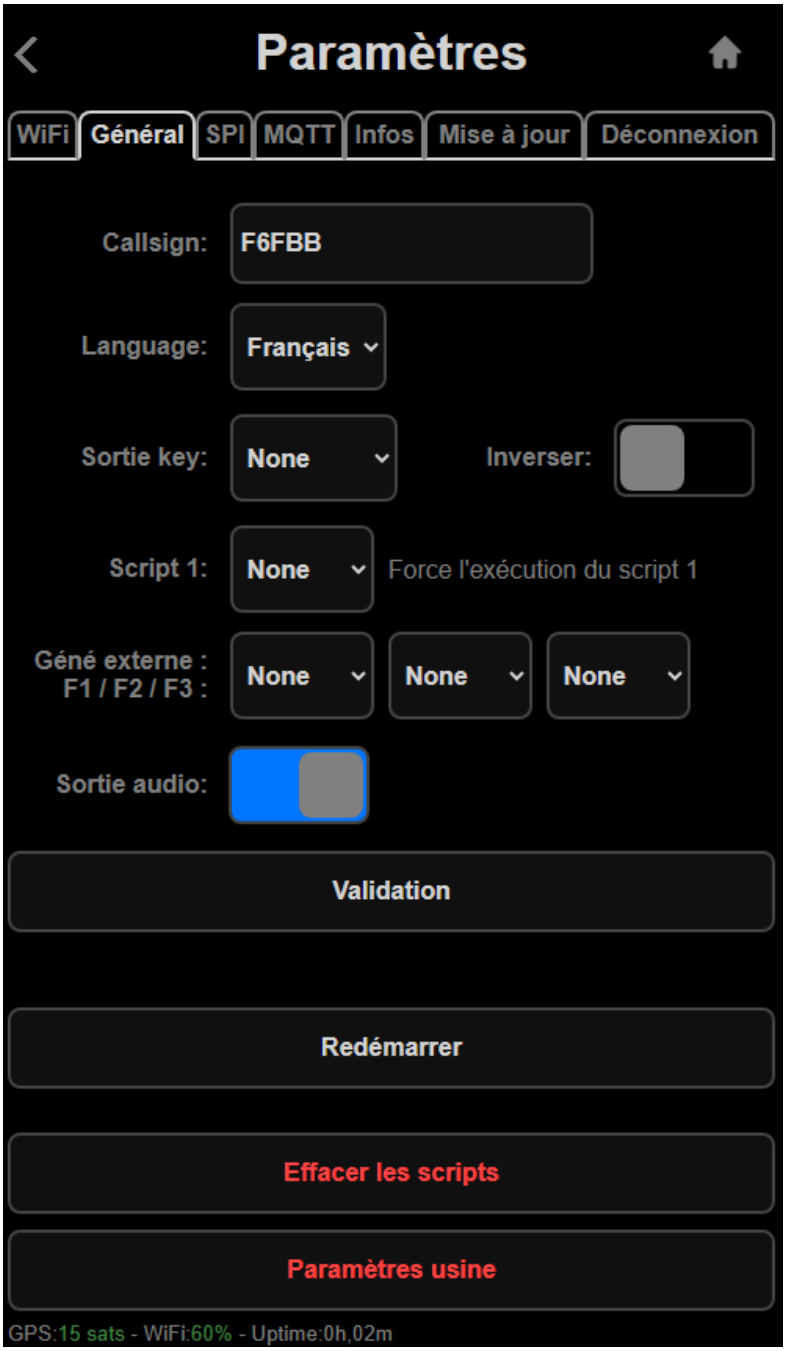
Access through an Internet WiFi network

- If a WiFi network and its password are specified, FbBalise will try to connect to this network. If it fails, it starts its own WiFi access point.

Configuration access with user name / password

- If these fields are filled in, a user name and a password will be asked for at login.

5.2. General settings



Callsign

- Default callsign of the beacon

Language

- Interface language

CW KEY signal output (active if SPI is OFF)

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

CW KEY signal inversion

- Inverts the CW output signal.

1PPS output (active if SPI is OFF)

- Start-of-second pulse generated by the GPS (spare 1 to 6).

Script 1

- Selects a spare that will lock the beacon on script 1.

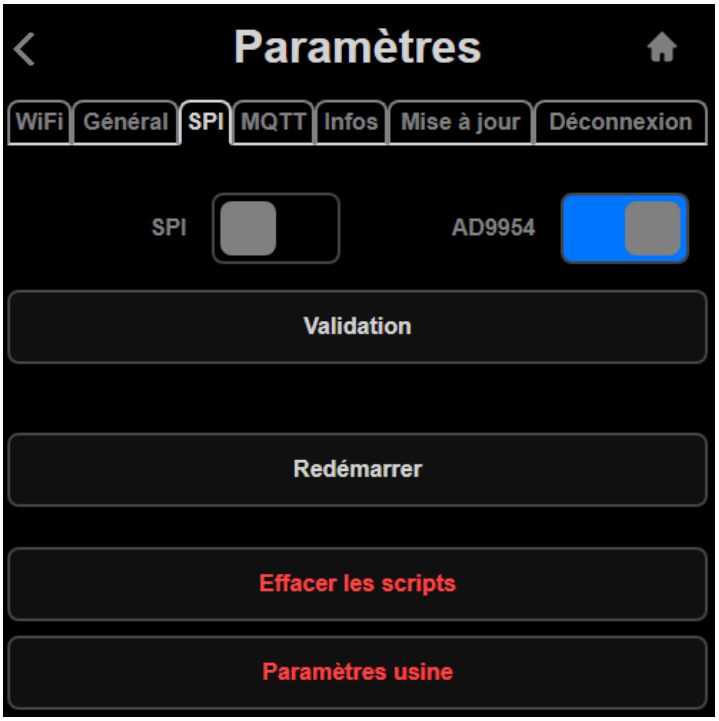
External generator

- specifies three spares giving the phase used to drive an external generator. See section External generator.

Audio output

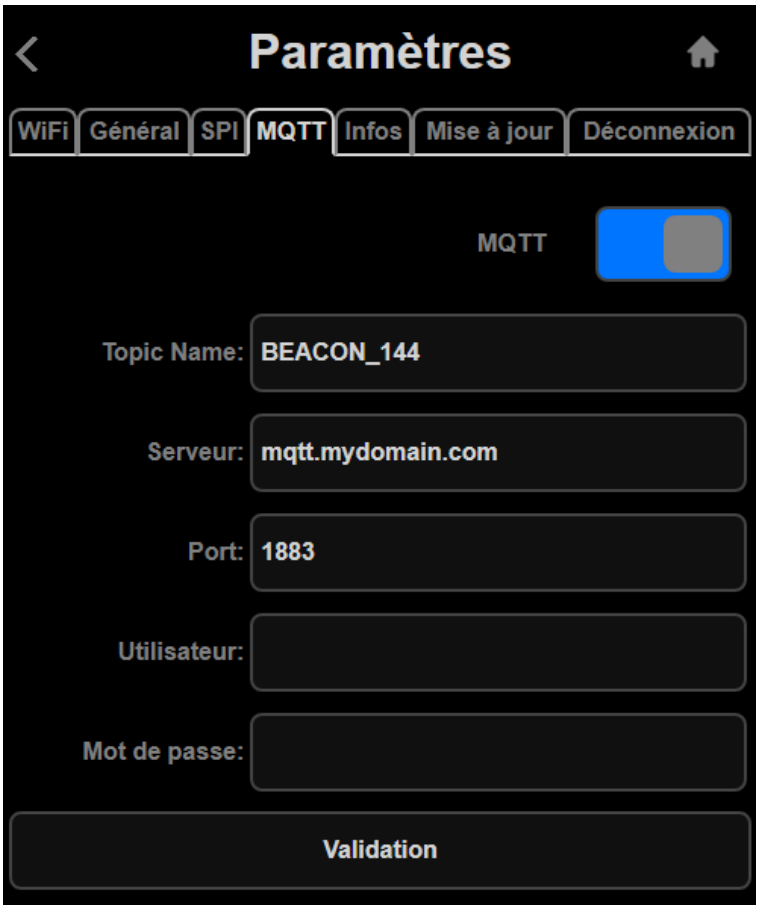
- Enables or disables the generation of the audio signal. A restart is needed for the change to take effect.

5.3. SPI configuration

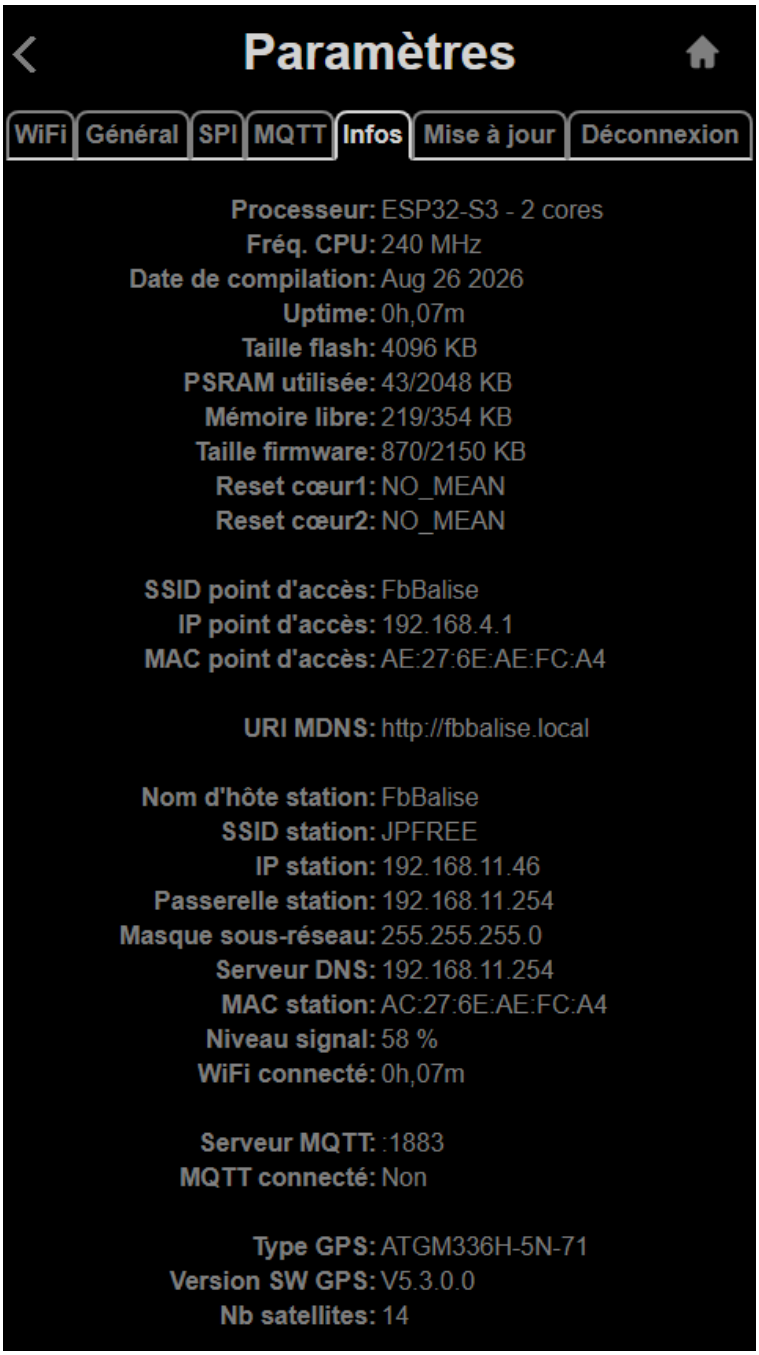


If both the SPI and AD9954 switches are enabled, the SPI configuration page is used to program an AD9954 DDS. See §12.

5.4. MQTT


Paramètres
WiFi
Général
SPI
MQTT
Infos
Mise à jour
Déconnexion
MQTT
Topic Name:
BEACON_144
Serveur:
mqtt.mydomain.com
Port:
1883
Utilisateur:
Mot de passe:
Validation

5.5. Infos



Gives general information about the system.

5.6. Update



This page is used to update the software with a new version. The file has the form “file_name.bin”.

After selecting the file, press the “update” button. The file is sent to the beacon module to update it. At the end of the update, the module is restarted.

Do not disconnect the module during the update.

The configuration can also be saved to and restored from a JSON file.

5.7. Logout

Closes the session. The login and password will be needed to access the web interface again.

6. Script keywords

- A parameter between square brackets is optional
- An expression can be
  - a numeric value
  - a character string between double quotes
  - an expression between parentheses, containing expressions, variables, numeric values, parentheses and the signs +, -, /, *, %
  - a condition between parentheses, containing operations, variables, numeric values, parentheses and the signs >, >=, <, <=, =
  - a function: RND, INT, ROUND, ABS, MIN, MAX (see section “Functions”)

6.1. BREAK

- BREAK has no parameter.
- BREAK immediately ends the innermost LOOP: execution resumes after its ENDLOOP.
- BREAK is usually placed inside an IF. Outside a loop, it is reported as an error when the script is saved.

```
LOOP  # waits until the GPS receives at least 4 satellites
      IF ($_NBSATS > 3)
            BREAK
      ENDIF
      DELAY 1000
ENDLOOP
```

6.2. DEBUG / ECHO

- DEBUG has a text parameter shown on the home page, below the editing window.
- ECHO is identical to the DEBUG keyword but does not show the DEBUG + line number information.
- Use double quotes around the argument, as it may contain spaces.
- If the argument contains a variable, its content will be converted and shown as text.
- A variable can be formatted, see Formatting a variable.
- In the following example, an external voltage divider with a ratio of 5.12 is assumed. The variable will always be shown with one decimal.

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

6.3. DELAY

- DELAY has an expression parameter giving the pause time in milliseconds
- DELAY can also wait for an absolute time, in seconds of UTC time (synchronised by the GPS), with the word UNTIL:
  - DELAY UNTIL s: waits for the next second s of the minute ($0 \le s < 60$).
  - DELAY UNTIL s EVERY p: waits for the next time t (in UTC seconds) such that t modulo p = s ($0 \le s < p$). For example EVERY 120 for an even minute (WSPR), EVERY 3600 for the start of each hour.
  - s and p can be values, variables or expressions.

- Warning: without UNTIL, the value is in milliseconds; with UNTIL, it is in seconds.
- The behaviour is strict: if the target time has just been passed, even by a few milliseconds, the wait goes on until the next period. It is up to the script to keep the necessary margin.
- If the time is not yet synchronised by the GPS, a warning is shown and the wait uses the internal clock.

```
DELAY (EXPRESSION)
DELAY UNTIL (EXPRESSION) [EVERY (EXPRESSION)]
DELAY UNTIL 10.5  # waits for second 10.5 of the minute
DELAY UNTIL 1 EVERY 120# waits for second 1 of an even minute
```

6.4. HEX (to be implemented)

- Function converting a number or expression into a hexadecimal string. It takes two parameters 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 is limited to this length.

```
HEX((4.52*10), 12) RETURNS THE STRING “02D”
```

- This function does not keep the sign; this is left to the user. So in 12 bits, 4095 and -1 both return “FFF”

6.5. IF / ELSIF / ELSE / ENDIF

- IF or ELSIF have a parameter between parentheses defining the condition.
    - If the condition is true, only the IF/ELSIF part is executed. Only one IF/ELSIF is executed; if no IF or ELSIF is true, the ELSE is executed.
    - There can be several ELSIF blocks.
    - If the condition is equal to 0, only the ELSE part is executed.
- An ENDIF must always mark the end of the conditional test.
- If the condition cannot be evaluated at run time (unknown variable for example), it is considered false and the script carries on.

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

6.6. LOOP / ENDLOOP

- LOOP can have an optional parameter giving the number of loops
- If the number of loops is not specified, the loop is infinite
- An ENDLOOP must always mark the end of the loop.
- The number of loops can be a value, a variable or an expression. LOOP 0 does not run the content of the loop.
- If the number of loops cannot be evaluated at run time (unknown variable, negative value), the loop is not run and the script carries on.
- The BREAK keyword exits a loop before its end.
- LOOP and IF blocks can be nested up to 10 levels.

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

6.7. MODE

- MODE takes one or more parameters giving the options of the mode
  - CW (speed in words / minute) [tone]
    - Tone 800Hz by default
  - OPERA PERIOD [tone]
    - PERIOD among OPERA05, OPERA1, OPERA2, OPERA4, OPERA8, OPERA32, OPERA65, OPERA2H
    - Tone 1500Hz by default
  - OOK48 [tone]
    - Tone 800Hz by default
    - Requires FbBalise V3 (with the GPS 1pps signal)
  - PI4 Type [tone]
    - TYPE among PI4, PI4-80, PI4-96, PI4-120
    - Base tone 800Hz by default
  - JT Type [tone]
    - TYPE among
      - 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
      - JTTY
    - Base tone 800Hz by default for JT/Q65xx and 1500Hz for WSPR and JTTY
    - JTTY is limited to the AD9954 DDS for now (not supported with an LMX2571)
  - JT MSK144 [nb_of_frames]
    - No tone setting. The centre frequency is 1500Hz with a shift of +/- 500Hz.
    - A frame lasts 72ms; a 30-second transmission needs about 400 frames.



If the tone parameter is specified, it is used for the frequency shift (FSK) of the selected mode instead of the default value.

```
MODE CW (EXPRESSION) [(EXPRESSION)]
MODE OPERA OPERA05 [(EXPRESSION)]
MODE OOK48 [(EXPRESSION)]
MODE PI4 PI4 [(EXPRESSION)]
MODE JT JT4G [(EXPRESSION)]
```

6.8. RESTART

- No parameter needed,
- Restarts FbBalise. The configuration and the scripts are kept.

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

```
ENDIF
```

6.9. SEND

- SEND sends a text given as argument, using the mode previously defined by the MODE keyword.
- Use double quotes around 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 Formatting a variable.
- The content may be limited by the mode used.
- With OPERA, only a valid callsign is accepted as argument.
- With JT, the callsign and the QTH locator can be sent, type 1, 2 or 3 messages for WSPR, and free text (up to 80 characters) for JTTY.

```
SEND  "F6FBB JN12BT "
SEND  "TENSION $($_A2 * 4.1):1 " #  1 digit after the decimal point
```

6.10. SET

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

```
SET $NOMDEVAR ($AUTREVAR / 12.5)
SET $NOMDEVAR 12.5
SET $NOMDEVAR ($NOMDEVAR + 1)        # increments $NOMDEVAR
```

6.11. TUNE

- TUNE has an expression parameter giving the duration of the signal in milliseconds
- TUNE accepts the same absolute form as DELAY (UNTIL / EVERY, in seconds): the carrier is transmitted until the given time. See DELAY.

```
TUNE (EXPRESSION)
TUNE UNTIL (EXPRESSION) [EVERY (EXPRESSION)]
TUNE UNTIL 58      # carrier until second 58
```

6.12. TIMESTAMP

- TIMESTAMP has a text parameter giving the password used to compute the key. This password **MUST** be in upper case. 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 computation is the GMT time given by the GPS.
- If the time is not valid (i.e. no GPS), the message “TIME ?” is sent. A warning is then shown and the script carries on.

```
TIMESTAMP PASSWORD
```

7. Variables

7.1. Declaring and using variables

Variables are always numeric. They are defined or redefined with the “SET” keyword. A special variable **$(EXPRESSION)** needs no declaration; it contains the expression to evaluate and stands for its result.

If the variable is used in a “SEND” line, its value is formatted if requested and converted to text. The number of variables is limited to 10. Variables belong to each script: they are reset each time the script is started.

7.2. Predefined variables

Several system 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 write-only.

Variable	Value	Comment	R/W
$_I1 TO $_I6	0 or 1	State of the spare input 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	Voltage on the analog input 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	-1 if the GPS is not active	R
$_MINUTE	0 to 59 or -1	-1 if the GPS is not active	R
$_SECOND	0.000 to 59.999 or -1	-1 if the GPS is not active. With ms	R
$_DAY	1 to 31 or -1	-1 if the GPS is not active	R
$_MONTH	1 to 12 or -1	-1 if the GPS is not active	R
$_YEAR	yyyy or -1	-1 if the GPS is not active	R
$_UPTIME	Number of hours	Number of hours since start-up (float)	R
$_UPSCRIPT	Number of hours	Number of hours since the script started	R
$_LOOP	Number of loops	Number of iterations in the current loop	R
$_POWER	Voltage	Supply voltage of the module	R
$_TEMPE	Temperature	Or -1 if the BME280 is not connected	R
$_HUMID	Humidity	Or -1 if the BME280 is not connected	R
$_BAROM	Barometric pressure	Or -1 if the BME280 is not connected	R
$_ALTITUDE	Altitude (metres)	Or -1 if the BME280 is not connected	R
$_BALTI	Altitude corr. (metres)	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, password between square brackets.	R
$_NBSATS	>= 0	Nb of GPS satellites	R
$_SCRIPT	1 to 4	Returns the script or changes the current script. The requested script is checked before switching: if it is not usable, the current script carries on.	RW
$_TOD	0 to 86399.999	Seconds since midnight UTC	R
$_DOW	1 to 7	UTC day of the week: 1=Monday, 7=Sunday	R



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



The $_BALTI variable is 0 by default, so there is no pressure correction. It must be initialised with the altitude for the correction.



The $_TEMPE, $_HUMID and $_BAROM variables require a BME280 connected to the I2C port. The address is detected automatically.



The variables $_I1 to $_I6 and $_A1 to $_A3 set the spare port as an input. The variables $_O1 to $_O6 set the spare port as an output. If the spare port concerned is used for something else, this may affect its operation.



The voltage on a spare input must not exceed 3.3V, or the microcontroller will be destroyed. Higher voltages can be measured with an external voltage divider. The value can then be multiplied with an expression.

Examples:

```
SET $_O2 1   # Sets spare port 2 high

SET $_O2 0   # Sets spare port 2 low

IF ($_I1)    # True if spare port 1 is high

IF ($_A3 > 2.1)    # True if analog port 3 is above 2.1V

SET $_T0 "BATTERY VOLTAGE: $_POWER:2,V"# Sends the supply voltage to the Windows application on telemetry 0
```

7.3. Telemetry variables

When the beacon is installed with a GesBal, the GesBal periodically sends it (every 10 seconds by default) the measurements of its boards over the local network (UDP, port 5250). These frames do not leave the local network.

The measurement board matching the beacon is chosen in GesBal (Station / boards tab, “beacon” field of each board) by entering the MAC address of the beacon, shown in the FbBalise Infos tab on the “Station MAC” line. No setting is needed on the beacon. As soon as a first frame is received, the Infos tab shows a “GesBal:” line with the associated board, its measurements and the age of the data.

Variable	Value	Comment	R/W
$_GV	Volts	Supply voltage measured by the GesBal board associated with the beacon	R
$_GI	Amperes	Current drawn	R
$_GP	Watts	Peak forward power over the GesBal measurement period	R
$_GR	Watts	Peak reflected power over the GesBal measurement period	R
$_GS	1 to 999 or 0	SWR computed from the peak values. 0 if no transmission	R

Variable	Value	Comment	R/W
$_GT	Temperature	Board NTC1 probe (°C)	R
$_GU	Temperature	Board NTC2 probe (°C)	R
$_GT1 to $_GU3	Temperature	Same values for GesBal board 1, 2 or 3, whatever the board associated with the beacon	R
$_GBN	0 to 3	Number of the GesBal board associated with the beacon, 0 if none	R
$_GAG	Seconds	Age of the last data received from GesBal, 999999 if no data received	R



If a value is not available (no GesBal, no board associated with the beacon, probe missing), the variable is 0 and a warning is shown only once. The script carries on.



The $_GAG variable makes it possible to check that the data is recent before using it, for example IF ($_GAG < 60). The power and SWR values describe the last GesBal measurement period, i.e. the previous transmission.

Examples:

`SET $_T1 "U=$_GV:1V I=$_GI:2A P=$_GP:1W"`# Voltage, current and power to the Windows application on telemetry 1

`SEND "F6FBB $_GT:0C"` # Inserts the NTC1 temperature in a message

`IF (($_GAG < 60) & ($_GS > 3))` # True if the data is recent and the SWR is above 3

7.4. Formatting a variable

When a variable is used as text (in DEBUG or SEND for example), it can be formatted with the following format:

`$VAR:N,S`

- N is the number of decimals. The value is rounded according to this number.
- S is the separator character that replaces the decimal point.
- Formatting is optional, but must be given in 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 $_BALTI 400            # Barometric pressure corrected for the altitude (400m)
SEND "/_BARO$($_BAROM%100):0" # Sends the pressure 1016hPa: '/_BARO16'
```

8. 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 done 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 done first, from left to right.

Additions and subtractions are done next, also from left to right.

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

Examples:

(3 + 5 * 2) => 3 + 10 => 13

((3 + 5) * 2) => 8 * 2 => 16

(3 * 5 + 2) => 15 + 2 => 17

8.1. Functions

An expression can use the following functions. Their names can be written in upper or lower case.

- **RND(N)**: random decimal number between 0 (included) and n (excluded).
- **INT(x)**: integer part of x (rounded towards zero).
- **ROUND(x)**: x rounded to the nearest integer.
- **ABS(x)**: absolute value of x.
- **MIN(A; B)** and **MAX(A; B)**: the smaller / the larger of the two values.

The two arguments of MIN and MAX are separated by a semicolon, because the comma is accepted as a decimal separator (1,5 = 1.5). A function call may contain spaces without being enclosed in parentheses.

Examples:

SET $F (1400 + INT(RND(200))) # random audio frequency between 1400 and 1599 Hz

SET $D MAX(0; $D - 1) # decrements $D without going below 0

9. 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 reach. The result of a comparison is 1.0 if it is true, otherwise 0.0 if it is 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 exclusive OR	

Examples:

```
IF ($_A3 > 2.1)    # True if analog port 3 is above 2.1V
```

Runs if analog3 is above 2.1V

```
ELSE
```

Runs if analog port 3 is at or below 2.1V

```
ENDIF
```

```
IF ($_T & 2.1)     # True if the value of $_T is not 0
```

10. Writing a script

10.1. Creating the script

The script is typed (or pasted) in the editing window. When editing is finished, the save button sends the script and saves it in the ESP32 flash.

- The script is interpreted in upper case and makes no difference between upper and lower case.
- Each line must start with a keyword, followed by its arguments if any.
- Empty lines are ignored.
- The end of a line after the # character is ignored (except inside a quoted text).
- Indentation is automatic: there is no need to type spaces or tabs at the start of a line. The editor shows two spaces per LOOP and IF level, and places the cursor at the right level after the Enter key.
- The Tab key re-indents the whole script; pasted text is re-indented automatically. Spaces and tabs at the start of a line are not saved.
- While typing, the editor shows in orange an unclosed LOOP/IF block, an end of block without LOOP/IF, or a BREAK outside a loop.

10.2. Checking and autonomous operation

The whole script is checked, without being run, each time it is saved and each time it is started: syntax, keywords, number of parameters, modes, structure of the LOOP/IF blocks, expressions, functions and predefined variables.

- When saving, the script is always saved; the first error found is shown below the editing window.
- At start-up, a script with an error does not start. If a correct version of this script has already been saved or started, this last valid version is run, with a warning giving the error. So after a power cut, the beacon always transmits, even if a script being written was saved with an error.
- Once started, the script no longer stops on an error: a line that cannot be executed (unknown variable, undefined mode…) is skipped, its message is shown only once, and the script carries on.
- Conditions outside the script (GPS time not yet valid, spare port in use) are reported by a warning (“Warning”) and do not stop execution.
- Switching script with $_SCRIPT only happens if the requested script (or its last valid version) is correct; otherwise the current script carries on.
- Variables belong to each script and are reset each time it is started.

10.3. Simple script

This script will:

- Select CW mode at 15 words/minute, 800Hz tone
- Send the callsign and the 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
```

10.4. More advanced script

This script will generate:

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

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

10.5. Script with a conditional test

This script will generate:

- CW mode at 15 words per minute, 800Hz,
- three times the callsign, a 1s pause, then twice the QTH locator,
- then a 500ms pause, a 5-second tune and a 1-second pause
- Every 4 loops it sends the callsign in OPERA mode, then goes back to CW mode
- and will loop forever…

```
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
```

10.6. Script synchronised on the minute

This script sends the callsign 3 times followed by a one-second pause, then a tune until second 59, a 1-second pause, then the callsign in OPERA mode. It needs the GPS module for the time information.

```
MODE CW 15 800
LOOP
      LOOP 3
            SEND "F6FBB "
      ENDLOOP
      DELAY 1000
      TUNE UNTIL 59
      DELAY 1000
      MODE OPERA OPERA05 1500
      SEND "F6FBB"
      DELAY 1000
      MODE CW 15 800
ENDLOOP
```

10.7. Synchronised WSPR script

This script waits until the GPS receives at least 4 satellites, then, at each even minute, sends a WSPR frame on a random audio frequency, the callsign in CW, and a carrier until 2 seconds before the next even minute.

```
MODE CW 15 800
LOOP  # waiting for the GPS
      IF ($_NBSATS > 3)
            BREAK
      ENDIF
      DELAY 1000
ENDLOOP
LOOP
      DELAY UNTIL 1 EVERY 120# second 1 of an even minute
      MODE JT WSPR (1400 + INT(RND(200)))
      SEND "F6FBB JN12 23"
      MODE CW 15 800
      SEND "F6FBB"
      TUNE UNTIL 118 EVERY 120      # carrier until 2 s before the even minute
ENDLOOP
```

11. External generator

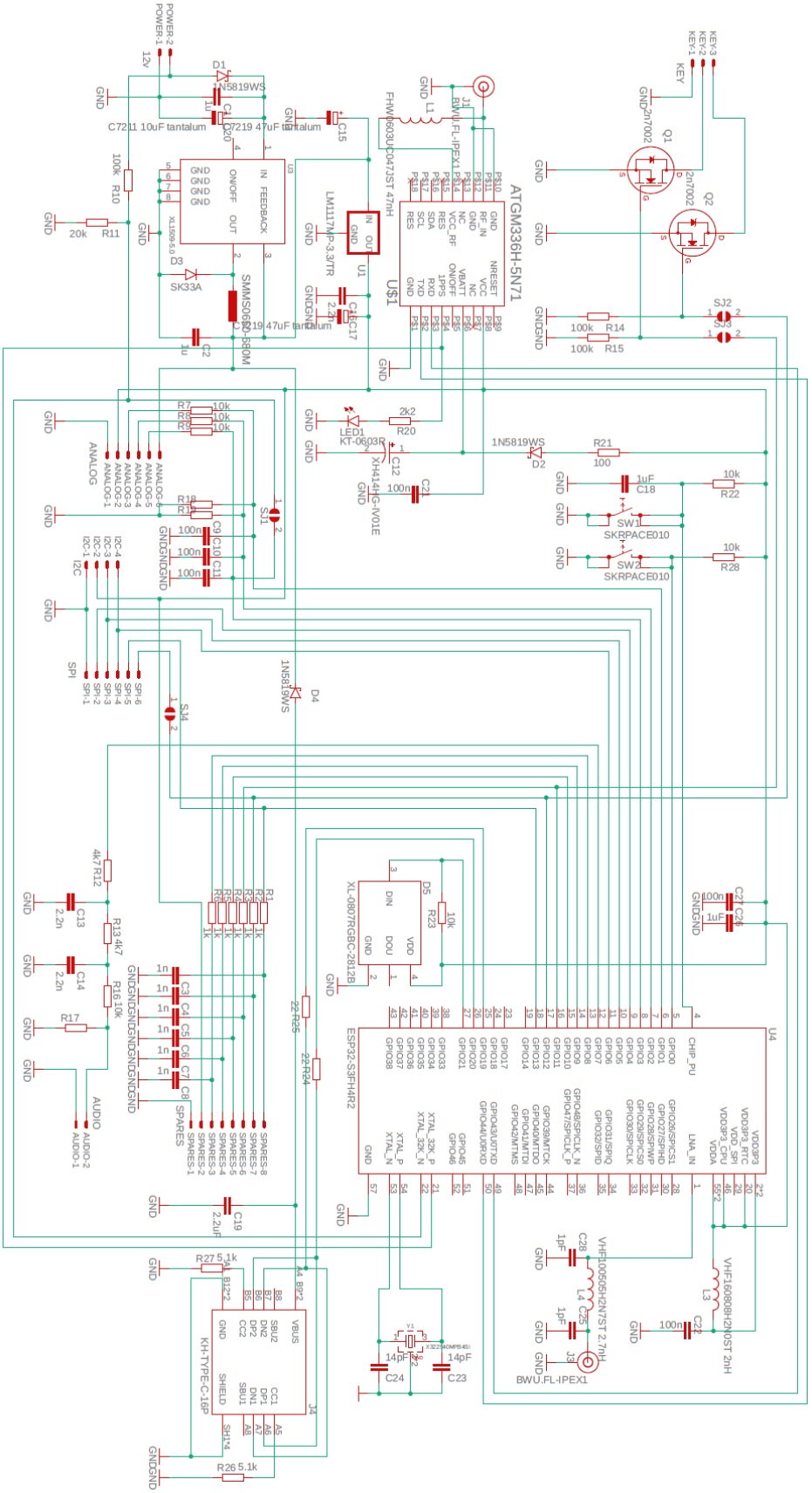
In the “Frequencies” tab of the configuration menu, 3 spares can be selected to drive an external OOK or frequency generator.

The level of the spares assigned to F1, F2 and F3 follows this logic:

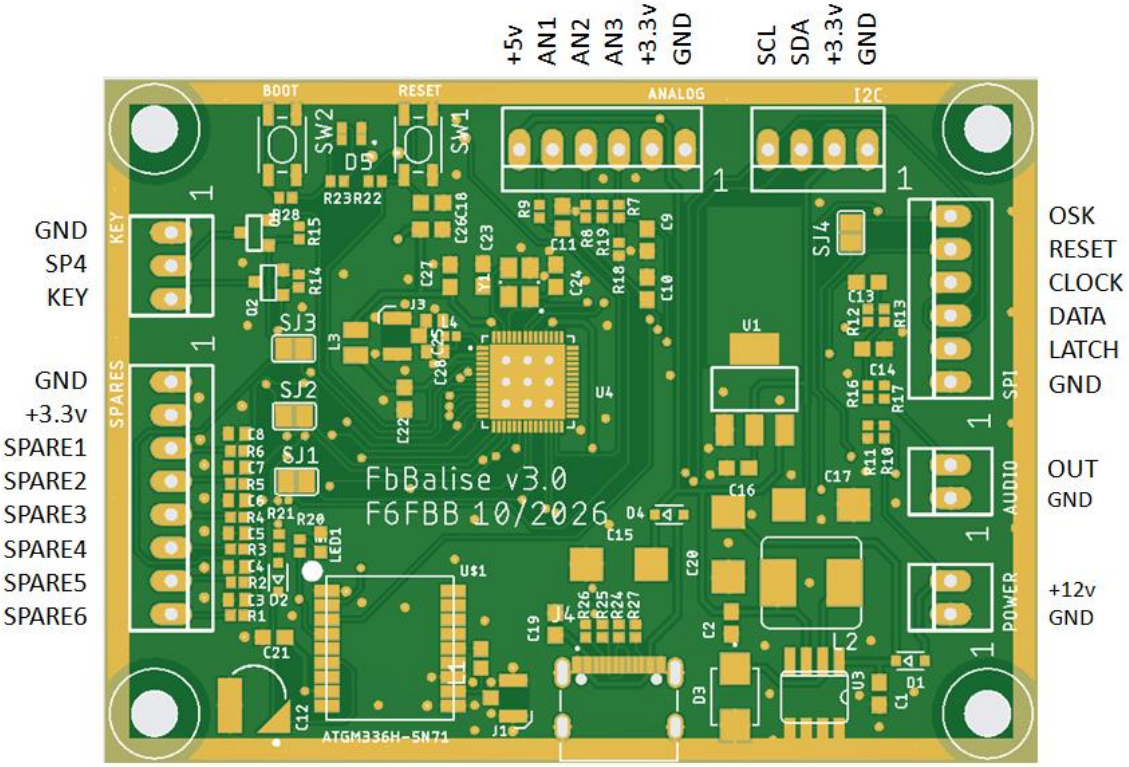
F3	F2	F1	Frequency to generate	
0	0	0	CW_OFF / SPACE	OOK generation or shift depending on hardware
0	0	1	CW_ON / MARK	OOK generation or shift depending on hardware
0	1	0	OPERA_OFF / SPACE	OOK generation or shift depending on hardware
0	1	1	OPERA_ON / MARK	OOK generation or shift depending on hardware
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 the 3 spares is not mandatory; all or some of them can be selected.
  - F1 => “KeyUp / KeyDown” for a CW-only beacon
  - F2 => CW / OPERA if PI4 is not used
  - F3 => PI4 mode

12. Schematic


POWER-2
POWER-1
12v
D1
1N5819WS
C1
1u
C7211 10uF tantalum
C7219 47uF tantalum
C15
GND
100k R10
20k R11
U3
GND
GND
GND
GND
ON/OFF
OUT
FEEDBACK
IN
D3
SK33A
SMMS060-2-BROM
C2
1u
C7219 47uF tantalum
LM1117NP-3.3/TR
U1
IN
GND
OUT
C16
2.2n
C17
GNDGND
FHW0603UC047JST 47nH
L1
J1
BWU.FL-IPEX1
ATGM336H-5N71
U51
PB10
PB11
PB12
PB13
PB14
PB15
PB16
PB17
PB18
GND
RF_IN
GND
NC
VCC_RF
VBATT
VCC
1PPS
RXD
TXD
GND
NRESET
VCC
PB9
PB8
PB7
PB6
PB5
PB4
PB3
PB2
PB1
SCL
SDA
RES
ON/OFF
KEY
KEY3
KEY2
KEY1
GND
2n7002
Q1
2n7002
Q2
GND
GND
GNDGND
100k R14
100k R15
SJ2
SJ3
GND
R7
R8
R9
10k
10k
10k
ANALOG
ANALOG-6
ANALOG-5
ANALOG-4
ANALOG-3
ANALOG-2
ANALOG-1
GND
R18
R19
SJ1
GNDGNDGND
C9
C10
C11
100n
100n
100n
I2C
I2C-4
I2C-3
I2C-2
I2C-1
SPI
SPI-6
SPI-5
SPI-4
SPI-3
SPI-2
SPI-1
SJ4
D4
1N5819WS
GND
2k2
R20
LED1
KT-0603R
C12
100n
C21
XH414HG-NV01E
GND
1N5819WS
D2
R21
100
1uF
C18
10k
R22
SW1
SKRPACE010
SW2
SKRPACE010
10k
R28
4k7 R12
GND
2.2n
C13
R13 4k7
2.2n
C14
R16 10k
R17
GND
GND
GND
AUDIO
AUDIO-2
AUDIO-1
GNDGNDGNDGNDGNDGNDGND
1n
C3
C4
C5
C6
C7
C8
SPARES
SPARES-8
SPARES-7
SPARES-6
SPARES-5
SPARES-4
SPARES-3
SPARES-2
SPARES-1
R1
1k
R2
1k
R3
1k
R4
1k
R5
1k
R6
1k
D5
XL-0807RGBC-2812B
DIN
VDD
GND
DOU
R23
10k
22 R25
22 R24
ESP32-S3FH4R2
U4
CHIP_PU
GPIO0
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO7
GPIO8
GPIO9
GPIO10
GPIO11
GPIO12
GPIO13
GPIO14
GPIO15
GPIO16
GPIO17
GPIO18
GPIO19
GPIO20
GPIO21
GPIO33
GPIO34
GPIO35
GPIO36
GPIO37
GPIO38
GPIO39
GPIO40
GPIO41
GPIO42
GPIO43
GPIO44
GPIO45
GPIO46
XTAL_P
XTAL_N
GND
GPIO26/SPICS1
GPIO27/SPIHD
GPIO28/SPIWP
GPIO29/SPICS0
GPIO30/SPICLK
GPIO31/SPIQ
GPIO32/SPID
GPIO47/SPICLK_P
GPIO48/SPICLK_N
GPIO38/MTCK
GPIO40/MTDO
GPIO41/MTDI
GPIO42/MTMS
GPIO43/U0TXD
GPIO44/U0RXD
VDD3P3
VDD3P3_RTC
VDD_SPI
VDD3P3_CPU
VDDA
LNA_IN
GND
GND
GND
GND

13. Connectors and jumpers



SJ1: supply voltage measurement on AN1

SJ2: open-drain OOK output

SJ3: open-drain PTT output

SJ4: SPI OOK output

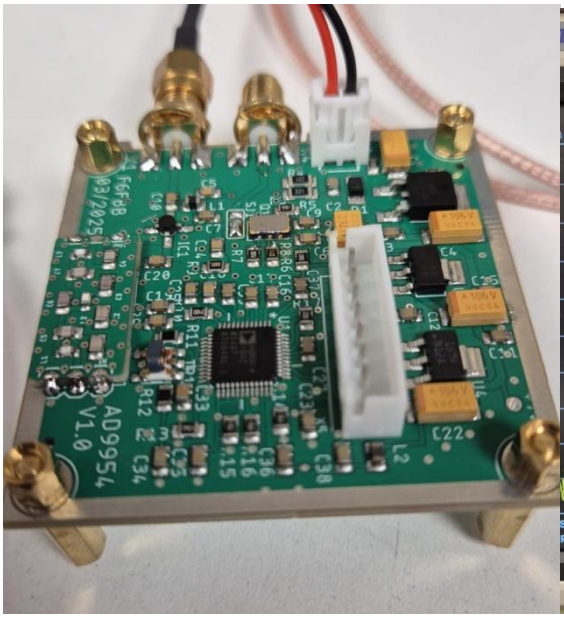
J1: GPS antenna

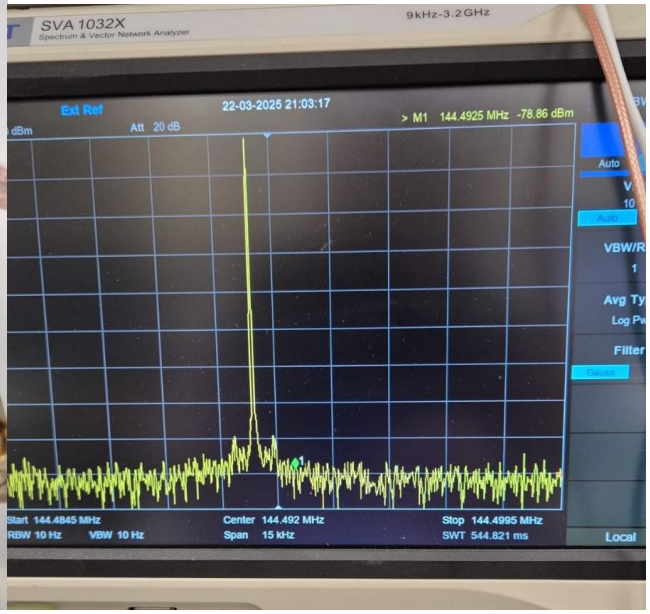
J3: WiFi antenna

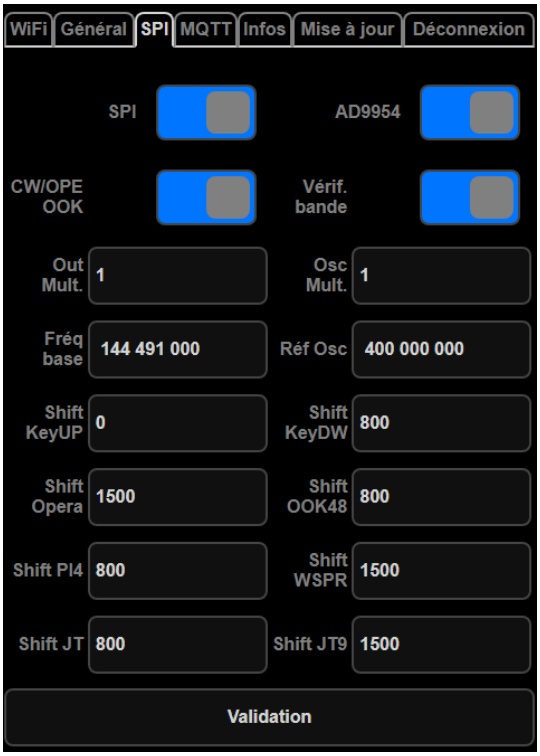
14. AD9954 DDS interface and board

A board based on the 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 is used to configure the chip and generate the RF signal in FSK or OOK depending on the modes and settings.





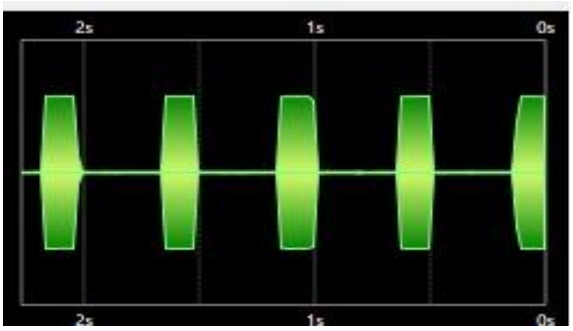


The actual frequency of the beacon must be entered. It is the DDS frequency multiplied by Out Mult.

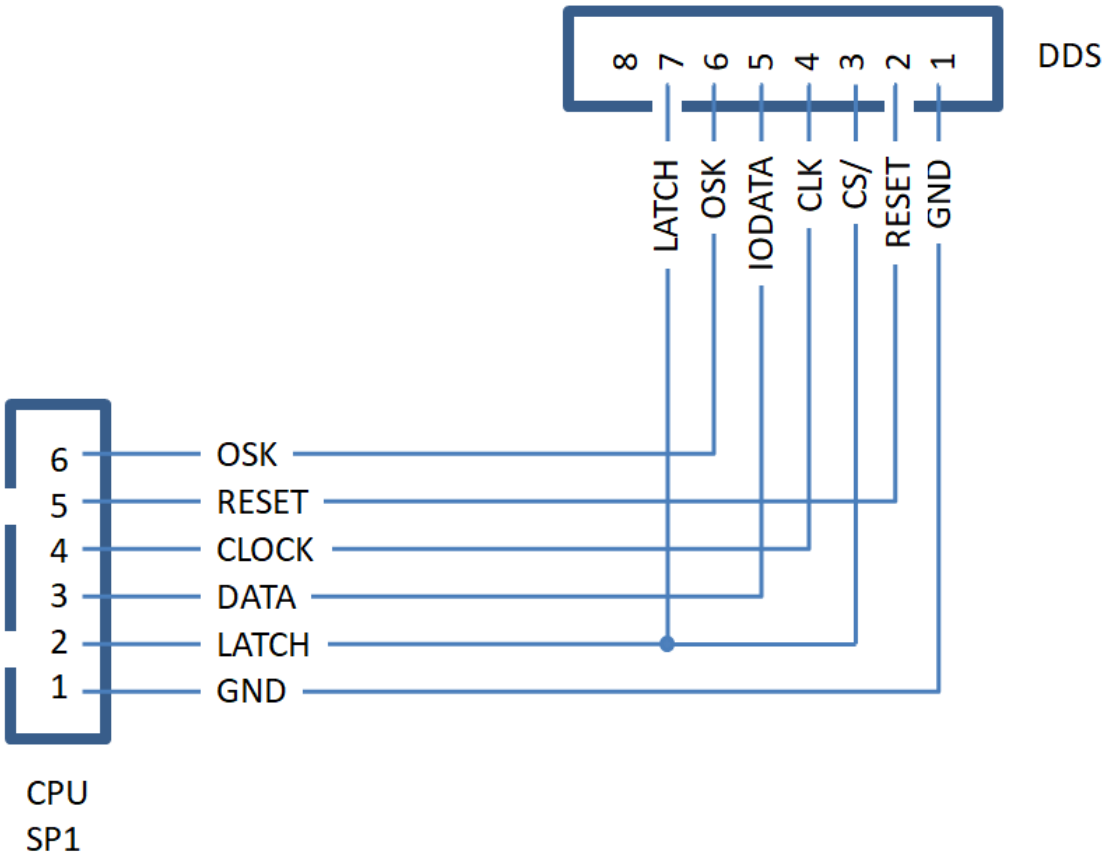
All shift and frequency values are in Hz.

“Osc Mult.” (1 or 4 to 10) sets the multiplication of the reference frequency to approach “Ref Osc” (400MHz by default).

“Out Mult.” is used to compute the DDS frequency from the beacon frequency and to scale the shift values of the modulation modes to a multiplication of the generated frequency. E.g. when 144MHz is generated and multiplied by 3 to produce 432MHz, the 315Hz shift in JT4G is reduced to 105Hz, so that the nominal value is obtained after the multiplier stages of the transmitter. The other FSK modes work the same way.



OOK modulation with “click-free” edges



FbBalise to DDS connector cable

15. Acknowledgements

The software is my own design. It runs on an FbBalise board and can use a GPS daughter board (for time setting and location).

The software is written in C/C++ in the “PlatformIO / VS Code” environment and includes several third-party ESP32 libraries.

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

Jean-Paul ROUBELAT, F6FBB.