Build a tune button for Yaesu transceivers (By PA0LUX)

18-9-2025: Added support for Tuning in Split Operation (VFO-B and VFO B). Originally only tuning VFO A was supported. ONLY for FM-N solution and ONLY for FTdx101D and FTdx101MP.

02-01-2024: Changes in text because this design can also be used for other Here Yaesu Transceivers without any modification.

30-11-2023: Note that the RS232 connection must not be assigned to a different function in the transceiver (i.e. Tuner)

01-04-2023: Corrected schematic for missing ground connection.

Transceivers which are really tested by myself with this design: FTdx-101 (D and MP), FT-991a

Transceivers which are checked by their CAT commands to be supported: FTdx-10, FT-991 (no split)

Most modern Yaesu transceivers will work with this design. The only criterion is that the transceiver’s CAT commands should be as used in the .ino file. And Mode FM- is available in the TXCR (unless TxREq is used).

Reason for this design: The FTdx-101 does not have a separate power setting when the tune signal is engaged. The tune signal (TxReq) is available on pin 11 of the linear connector. When this pin is connected to ground the transceiver outputs a carrier which can be used i.e., to tune an amplifier. Unlike other Yaesu transceivers it will output with the present power setting. There are several work-arounds but, I think, no one as nice as a one button tune would be.

You can find circuits and sketches with an Arduino-Nano (I used a clone) that will take care of one-button tuning. Probably every type of Arduino will work, as long as it has the Rx and Tx connections available.

The tune circuit is to be connected to the RS232 connector of the transceiver (thru a null modem cable), thus leaving the USB connector of the transceiver available for other logging and/or CAT software. The Arduino cannot directly be connected to the transceiver. The transceiver talks RS232 and the Arduino talks TTL levels. Therefore, a simple Rs232 – TTL level converter is used, widely available for about 2 euros. It comes complete with a 9 pin Rs232 D-connector.

The circuit has one momentary pushbutton and (optional) a red and green led. Pressing the button will result in a tune signal and when released the tuning signal will stop. The present power setting is restored after the button has been released. The tune power can be adapted in the sketch with the variable; **set\_tune\_pwr "PC020;"** The 020 means 20W. You can change it to anything between 005 and 100.

The leds indicate if there is communication with the transceiver. The green led turns on when there is an answer received from the radio. It does check for correct baud rate. When not connected or when the transceiver is powered off (and 13,8 V still present), or a wrong baud rate, the red led will be on and the green led will be off.

There are two different solutions available. The first uses the TxReq signal on the linear connector to produce an output signal. You will then need an Rs232 cable (3 wires; Rx, Tx and Gnd) and a 2x2-wire cable (TxReq + GND and 13,8V + GND) to the linear connector. The 2 x GND is because you could also use another source for the 13,8 V.

In the schematic you can see that output 13 of the Arduino is driving a transistor, which in turn drives a small 5V reed relay. The contact of the relay shortens pin 11 and 15 on the linear when the tune button is pressed. For transceivers other than the FTdx101 it is possible that you have to use a different connector/pin for the TxReq signal.

The second solution does not use the TxReq signal. Instead of that it uses the FM-N mode. At tune button press, it not only stores the present power setting but also the present mode. It will then engage a FM-N transmission. After the button has been released it restores mode and then power.

I first used AM here, but found that when an AM transmission was started, there was a short increase (peak) in output power. I do not know if that is common for all Yaesu transceivers, but switching to FM-N solved that issue.

Both solutions work fine. FM-N is recommended, it easiest and it supports Split mode.

It is advised to use a connector at least at one end for the connections between Arduino and Rs232 level converter.

Note; Do NOT connect USB and RS232 (of the Arduino) at the same time, it will not work (will do no harm either). Also, better not to connect 13,8V and USB at the same time.

To program the Arduino is very simple. Do not program while the Rs232 level converter is connected.

Make sure you have the Arduino IDE (downloaded and-) installed on your PC. This is the editor for sketches. Connect your Arduino to the PC and check which new port has just shown up in your computer. Now in the Arduino IDE choose via “Tools” menu this port and choose the correct Arduino board (in my case Nano).

Copy one of the sketches (. ino file) below to your PC and create a folder with exactly the same name as the copied file. Now move the copied file into the new folder.

Now in the Arduino IDE go to “Files” menu, choose open and navigate to the .ino file and open it.

Press the “right arrow” in the IDE. The program will be uploaded to the Arduino. It will indicate when ready.

Set the Rs232C rate in the transceiver to 9600 bd. Better not to set it higher. Connect the cables and your tune button should work now.

The interior and exterior:

