Frequency counter with a PIC and minimum hardware

by Wolfgang "Wolf" Büscher, DL4YHF
Last updated: 2015-07-29.

to main site

This document describes the construction of small frequency counter with a cheap PIC microcontroller and a few seven-segment LED digits. The main features of the frequency counter are:

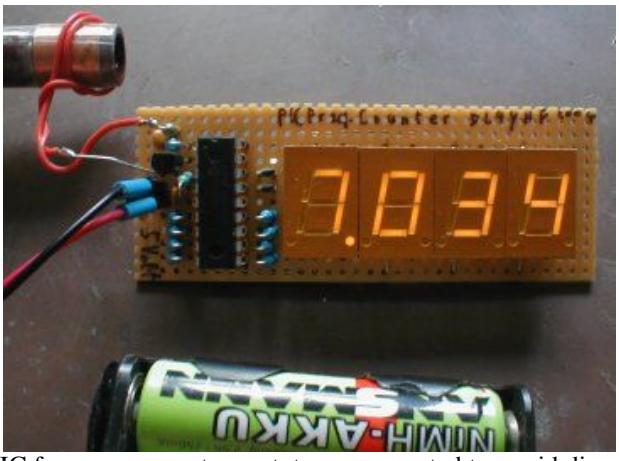
- frequency range 1 Hz ... 50 MHz (prototype worked up to 60 MHz but this exceeds the PIC's timing specifications)
- four or five digits resolution (display for example x.xxx kHz, x.xxx MHz, or xx.xx MHz); or 6 digits with F8FII's modification
- automatic range switching with different gate times
- optional addition or subtraction of a frequency offset (programmable)
- optional preamplifier for the input signal
- can be built on a single-sided 'breadboard-style' circuit board
- firmware available for common-cathode as well as commom-anode displays
- very low component count: a PIC 16F628,
  4 or 5 7-segment LED displays,
  a 4- or 20-MHz crystal and a few resistors,
  optionally one transistor and a few diodes to drive the 5th digit,
  optionally one other transistor for the preamplifier.
- since 2006-05: preprogrammed frequency offset for transceivers with 4.0 MHz IF
  (see the author's experiences with "Miss Mosquita", a 40 meter QRP transceiver)
- optional (configurable) power-saving mode which automatically turns the display off
  if the frequency didn't change significantly within 15 seconds

Contents of this document:

The PIC firmware for the frequency counter can be downloaded from this link (includes hex file for all display variants and the assembly sourcecode).
A PIC programmer which you need to program your PIC 16F628 is available on DL4YHF's website. Please do not ask me to send programmed PICs or even circuit boards to you, my spare time is too short - and you will find anything you need on these "do-it-yourself" pages ! If your counter shows a strange frequency initially, enter setup mode to set the frequency offset to zero (it sometimes happens that PIC programmers don't erase the EEPROM where the frequency offset is stored).

Construction - Variant 1

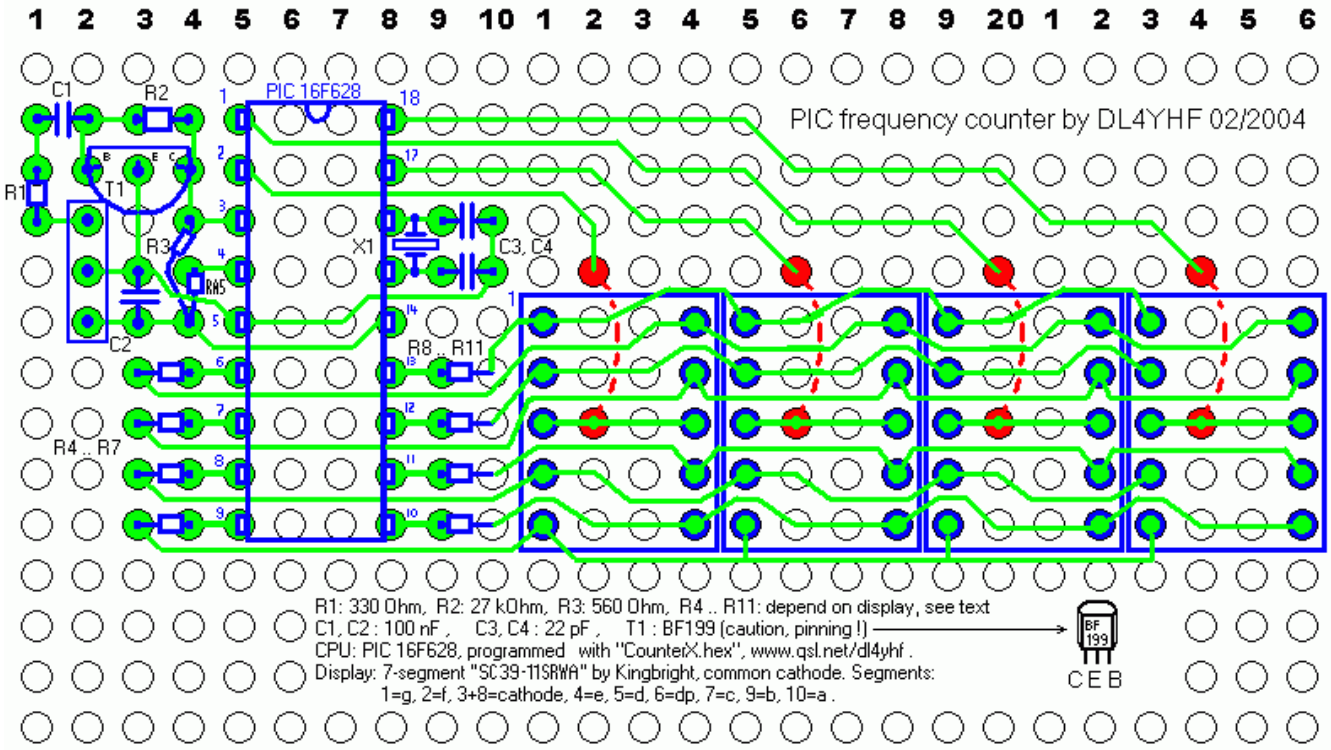
The first prototype with very old, low-efficiency 7-segment displays looked like this:



(PIC frequency counter prototype, connected to a grid dipper)

A flashing decimal point indicates a frequency in kilohertz, a steady point indicates a frequency in Megahertz - which is more common for the intended use in dip meters and QRP transceivers.

The circuit diagram is so simple that I didn't make a complete Eagle schematic for it yet (other builders did, see links). The first prototype was built on a breadboard with round pads. A very similar design could be used for a single-sided printed circuit board. Four bridges (drawn in red colour below) must be soldered on the board before fitting the seven-segment displays. Note that the crystal is mounted on the bottom side ! Some of the display signal are routed with thin enamelled copper wire, especially those between two pads.



(suggestion for building the counter on a breadboard. Use firmware "counter1.hex" for this circuit, and a 4-MHz crystal)

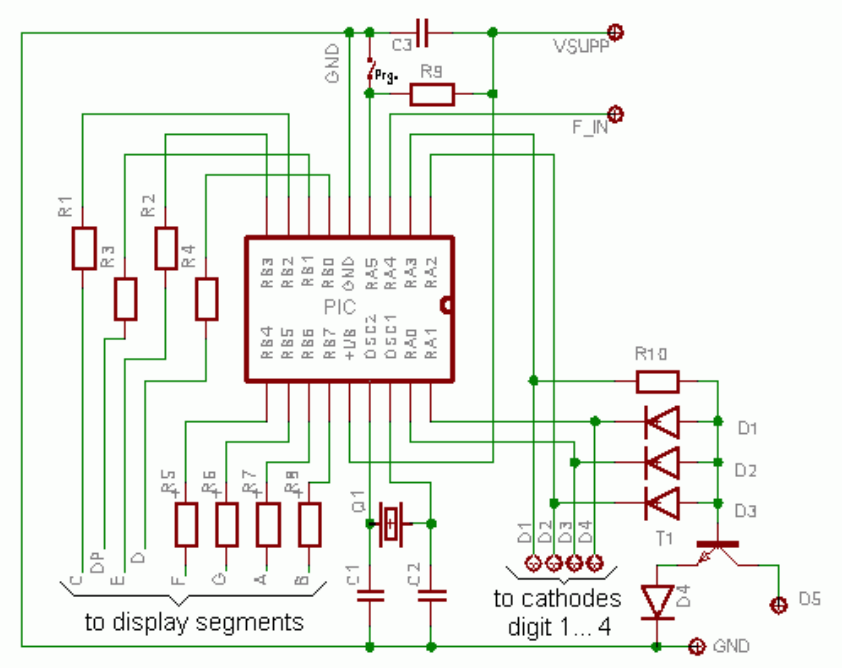
The resistors R4..R11 set the brightness and the power consumption. If you find some suitable low-current displays, use 1 kOhm or even more. Very old displays require more current, use 390 Ohms in that case. The PIC's output ports can source and sink 20 mA per pin, so the *digit* drivers are a slightly overloaded with 390 Ohm resistors *per segment* when displaying "8.". If you don't have to care for power consumption and want to use old displays with high current, use four PNP (!) transistors in the cathode(!) drivers as non-inverting emitter followers, no base resistors are required (base to PIC, collector to ground, emitter to common cathode).

For the displays "SC39-11SRWA" by Kingbright it is unnecessary to use driver transistors, these high-efficiency displaye are bright enough with 1 kOhm current limiting resistor per segment. In Germany you can -or, at least, could- order these displays at Reichelt, look for "SC 39-11 RT" in their catalogue (in fact, these used to be SC39-11SRWA displays; aka "super bright" / high-efficiency display. Do NOT use the low-efficiency types like SC39-EWA. You will be disappointed by a dark display, at least with the resistor values stated above).
Beware, other distributors of electronic components (like C..) sell such displays for much higher prices !

Construction - Variant 2

For the second prototype, a 5-digit display was made on a single layer PCB shown below. The segment resistors were used to connect the display board with the PIC board (which is usually mounted in a 90° angle behind the display). To make connection between PIC and display board easier, a second firmware variant was created with slightly modified pin assignment. On this occasion, the PIC's clock frequency was increased to 20 MHz to have a better resolution at higher frequencies.

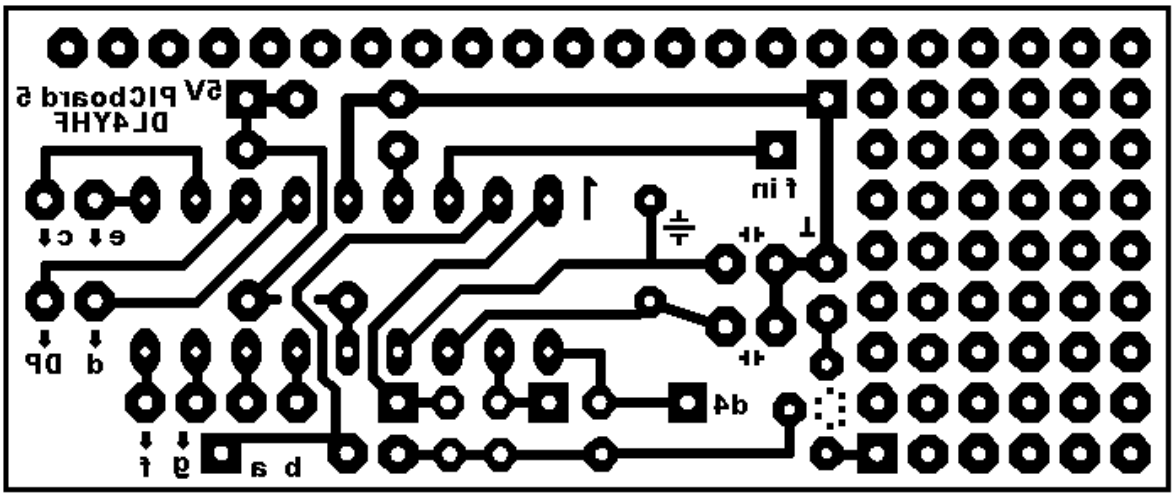
Schematics of the PIC board (display variant 2)



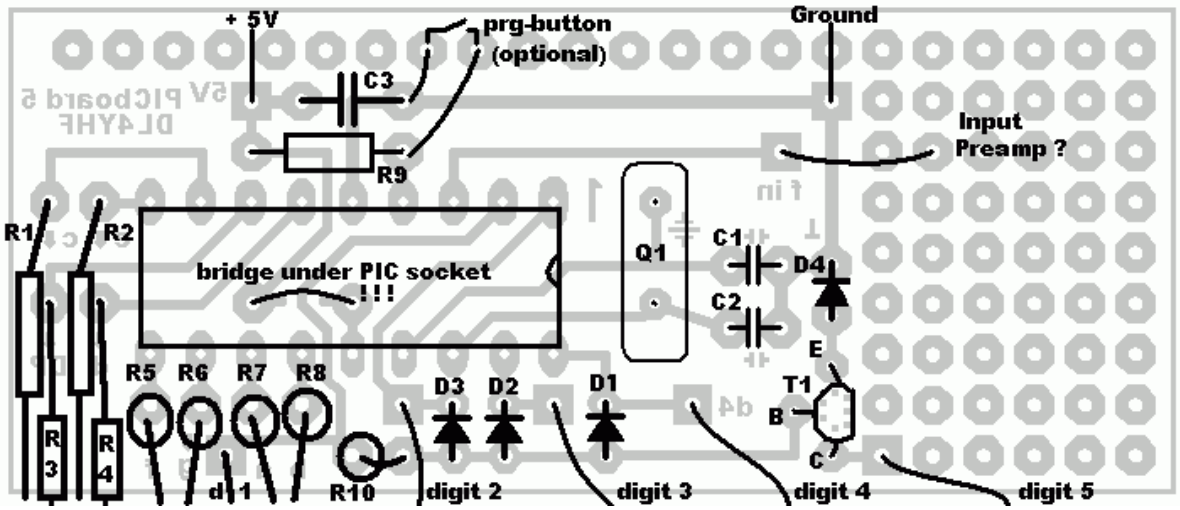
Component values for variant 2 (as shown above): R1...R8 = 1k, R9=R10=10k, D1..D4=1N4148, T1=BC547 or similar, C1=C2 about 22pF (select them to tune the crystal to exactly 20 MHz), C3=100nF, Q1=20 MHz, PIC: 16F628 programmed with firmware "counter2.hex".

Layout of the PIC board (display variant 2). The board contains the decoder for the 5th digit, and some breadboard area for the preamplifier (if needed). Print this image with exactly 300dpi, which can be achieved with IrfanView and other utilities. The image is 700 * 300 pixels large, so the board dimensions will be 2.33 * 1 inch, or 5.93 * 2.54 cm (check this *before*

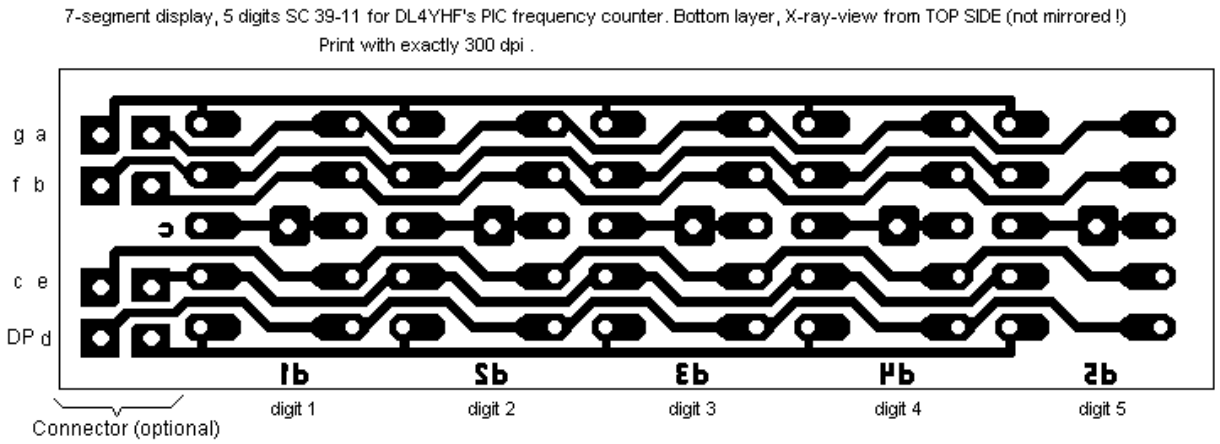
etching..). The display is mounted on the lower side of the PIC board as shown below. The square pads near the display are the cathode connectors (from left to right: d1..d4 are outputs directly from the PIC, d5 is driven by transistor T1).



Placement of the components on the PIC board (Ridiculously simple, isn't it ? But don't forget to place the bridge under the PIC socket before soldering !).

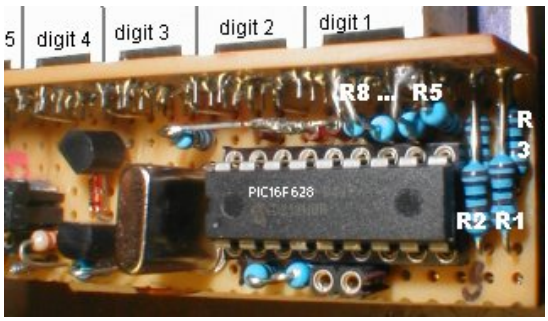


Here is one suggestion for a layout for the display board (display variant 2, also print this with *exactly* 300 dpi) :



This board is quite universal, I plan to use it for a PIC-based DDS generator too. For the frequency counter, the connector pads "g","a","f","b" are not used because these four resistors R5..R8 are connected from the PIC board directly to the *backside of the first seven-segment digit* (see below). The letters on the left side are the segments. The 5 cathodes are not routed to the side of the display PCB - it was impossible with such a small single-layer PCB. Use short pieces of wire to connect the square cathode pads on the display board to the square pads d1...d5 on the PIC board.

To save space on the front panel, the connector between PIC- and display-board can be omitted, and resistors R5..R8 on the PIC board soldered to the pins of the leftmost 7-segment display (digit 1). Resistors R1..R4 are soldered to the lower 4 connector pads on the display board. See photo of the second prototype, with the PIC still mounted on a raster board:



Well, using the resistors to connect the CPU board with the display isn't really nice - feel free to do better (others did - see links).
BTW this prototype perfectly fits in a miniature QRP transceiver called Miss Mosquita !

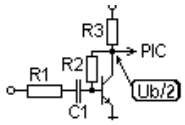
Common-cathode or common-anode display ?

The two display variants (COUNTER1 and COUNTER2) described above are for 7-segment displays with common cathode (CC). But the downloadable software achive also contains a third firmware file (COUNTER3.HEX) for common-anode displays (CA).

COUNTER3 uses the same pins as COUNTER2, but the control outputs are inverted. In the circuit, use PNP transistors for the CA display instead of NPN for T1 to drive the 5th digit, furthermore connect D1..D4 with reverse polarity, and connect D4 to Vsupp (positive supply voltage) instead of GND.

Preamplifier

The preamplifier consists of a single HF silicon transistor. I used a cheap BF199 from the junk box, it works well up to 30 MHz and with reduced sensitivity up to 50 MHz. Make the connection from the



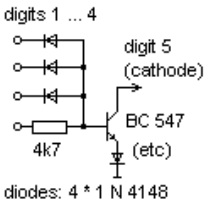
collector to the PIC counter input (pin 3 = T0CKI) as short as possible, because every pF of capacitance counts.

With a BF199 in the preamp, R2 is 27 kOhm (+/-), and R3 must be as low as 560 Ohm to achieve the necessary bandwidth. The DC voltage at the collector and the voltage across R3 should almost equal - if not, adjust R2.

To save area on the breadboard, you can leave away the preamplifier. If you want to feed the counter with a TTL signal, leave the preamplifier away, which saves another 4 mA. If the maximum frequency in your circuit is below 10 MHz, you may increase the value of R3 and R2 by the same factor (say R3=1.2 k, R2=56k) to save some current when using the counter in a battery-powered device. R1 sets the input impedance and also the sensitivity. With R1=330 Ohm, the prototype required an input voltage of 600 mVpp (peak-to-peak) at 40 MHz and 150 mVpp at 15 MHz. If you need a higher input resistance, add a FET buffer before the bipolar transistor. Or use a fast integrated comparator as input stage if you find one in your junk box.

Added 2006-04: You will find a preamp with high input impedance on the pages of the DL-QRP-AG - see links !

Driving the fifth digit (optional)



The PIC firmware always drives a fifth digit as the least significant digit, if the measured frequency is above 10 kHz. Because there was no free output pin available on the PIC 16F628 to drive another digit, a logic combination from the first four digit multiplexer outputs was used (digits 1 ... 4).

When all digit multiplexer outputs are passive (digits 1...4 = high), the optional fifth digit is driven. The schematics on the left side show one suggestion for a simple 'decoder' for the fifth digit, which basically is a four-input NAND gate realized with a few diodes, one resistor, and an NPN transistor.

The diode between the emitter and ground is used because without it, the transistor may conduct unwantedly if its base-emitter threshold voltage (typically 0.5 to 0.6 V) was less than the forward voltage of the other diodes in this stage (which will be about 0.6 to 0.7 V for a 1N4148). Of course one could use Schottky diodes with lower forward voltages to eliminate the emitter diode, but 1N4148 are more likely to be found in a junkbox than BAT41's (or similar).

If you don't need it, leave the 5th digit away to save some current and a few components. For this reason, the 'single zero' (if no input signal is present) is not displayed in the 5th, but in the fourth digit.

Power consumption

The prototype (with R4..R11 = 390 Ohm) drew an average supply current of 40 mA (with 5 low-efficiency digits). With high-efficiency or low-current displays this can be significantly reduced. With 1kOhm segment resistors, the display draws a total current below 20 mA (with 5 digits "SC39-11"). The PIC itself draws about 4 mA at 20 MHz, and less than 1 mA when running at 4 MHz - so the preamp draws more current than the controller itself !

Display ranges

The display range is automatically switched to give the maximum readout accuracy (with 4 digits). The gate time is also selected automatically as listed in the following table:

Frequency range	Display	Gate time	Decimal point
0 ... 9.999 kHz	`x.xxx`	1 second	flashing (which means "kHz")
10 ... 99.99 kHz	`xx.xx(x)`	1/2 second	flashing
100 ... 999.9 kHz	`xxx.x(X)`	1/4 second	flashing

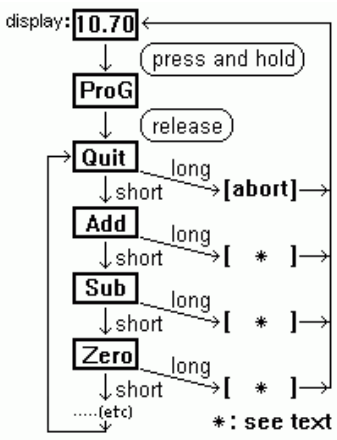
1 ... 9.999 MHz	x.xxx(X)	1/4 second	steady (which means "MHz")
10 ... 50.00 MHz	xx.xx(X)	1/4 second	steady

(On this occasion: "MHz" is Mega-Hertz, "mHz" would be milli-Hertz, but that's another story...)

Adding or subtracting an offset frequency

If the counter is used in a shortwave receiver or transceiver, you may want to add or subtract an offset value from the measured frequeny. The offset frequency is the same as the intermediate frequency in many cases, because the counter is usually connected to the receivers VFO (variable frequency oscillator). For this purpose, a programming mode (aka "setup mode") has been implemented in the firmware so you can enter an offset frequency without reprogramming (or even reassembling) the PIC firmware.

The signal RA5 (pin 4 of the PIC 16F628) will is used to switch from normal counter mode into programming mode. Usually the level on RA5 is high because it is connected to the supply voltage via pullup resistor (10k to 22k). If you will never need to add or subtract a frequency offset, connect it permanently with the supply voltage (there must be a defined level on RA5, unfortunately it has no internal pullup resistor). By pulling RA5 low (connect pin 4 and pin 5 of the PIC with a small screwdriver), the firmware will be instructed to use the currently measured frequency as the new offset value. In other words, you must *apply the offset frequency to the counter's input*, wait until the value is displayed correctly, and then enter the programming mode as explained below.



The program flow chart on the left shows how to enter programming mode, how to select a menu, and how to execute the associated function. To enter programming mode, press and hold the programming key (or connect pin 4 and 5 of the PIC with a small screwdriver), until the PIC shows "ProG" on the LED display. Then release the "key". You are now in the first menu of the programming mode.
To select the next menu, press the key for a *short* time (less than a second). To execute the selected function, press the key for a longer time (more than a second). The menu functions are :

- "Quit" : Aborts programming mode without changing anything.
- "Add" : Saves the previously measured frequency permanently, so it will be added in future.
- "Sub" : Saves the previously measured frequency permanently, so it will be subtracted in future.
- "Zero" : Sets the frequency offset to zero, so the display will show the measured frequency without offset. The previously programmed offset will be lost.
- "Table": Allows you to select a predefined offset value from a table. The table itself is also located in the PIC's data EEPROM, so you may find different values in it. When skipping through the table, the frequencies are shown in numeric form, like 455.0 (kHz), 4.1943 (MHz), 4.4336 (MHz), 10.700 (MHz). After selecting an entry (long keypress), you will be taken back to the main menu to select "Add" or "Subtract".
- "PSave" / "NoPSV": turns the power-saving on/off. In power-saving mode, the display is turned off after 15 seconds of no "significant" change in frequency, and on again as soon as the frequency changes by more than a few dozen Hertz (in the 3..4 MHz measuring range). Added in May 2006 for battery-powered equipment like QRP transceivers.

Note:

There may be more menu items than shown here, but the principle remains the same.

The frequency offset values are saved as a 32-bit integer numbers in the PIC's data EEPROM (at the EEPROM's first four memory locations, high-byte first, low-byte last). If you have no signal generator to produce the offset frequency for programming, or cannot tap the BFO frequency of your homebrew shortware receiver, you can enter the offset value with a suitable PIC programmer (like DL4YHF's WinPic). Use a scientific pocket calculator to convert the frequency (in Hertz, positive or negative) into a hexadecimal number, and enter this value in the PIC programmer's EEPROM DATA memory window. If you use WinPic, enable the HEX editor before typing the values into the memory window. Some examples:

```
4.194304 MHz : Add= 00 40 00 00  Subtract= FF C0 00 00  (yes, so simple)
4.433619 MHz : Add= 00 43 A6 D3  Subtract= FF BC 59 2D
0.455000 MHz : Add= 00 06 F1 58  Subtract= FF F9 0E A8
10.70000 MHz : Add= 00 A3 44 E0  Subtract= FF 5C BB 20
```

If the subtracted offset is higher than the counter's input frequency, the result of the subtraction is negative. The frequency counter makes the result positive before displaying it. This way, you can use the counter also in receivers where f_IF = f_RX + f_LO, or f_RX = f_IF - f_LO which means increasing LO frequency means decreasing RX frequency (the counter will seem to "run backwards" but that's no mistake).
Example for DL2YEO's 30 meter band QRP transceiver: f_RX = f_LO - f_IF = 14.314 MHz - 4.194 MHz = 10.120 MHz, which is the calculation inside the counter (f_LO=measured input, f_RX=display value, f_IF=programmed offset). If you don't need the 10-MHz-digit on the display, set the offset to -14.194 MHz instead of -4.194 MHz. This will give better display resolution, so you only need 4 digits (f_RX=10.120 MHz will be displayed as 120.0 kHz, which is sufficient because the receiver's tuning range is only 20 kHz anyway).

Some commonly used IF frequencies can be recalled from the "Table" menu, so you don't have to measure or enter them yourself. Please contact me if you are one of those QRP homebrewers and want to have other frequencies in the built-in 'standard IF table'.

How it works

Basically the program runs in an endless loop, with the exception of the initial lamp test, programming mode, and power-saving mode which are not explained here. In the main loop the following steps are performed:

1. Prepare a coarse frequency measurement for the automatic range switching: Program the asynchronous prescaler to divide by 64, so the highest external frequencies can be detected (theoretically 64 MHz, but this exceeds the PIC's specification).

2. Count the input pulses for 1/16 second, using the PIC's *TIMER0* module in counter mode. During this time, the display multiplexer keeps running. In fact, the counting loop takes exactly 50 microseconds, including the multiplexer routine. 1250 counting loops result in a gate time of 1/16 seconds.
   In the sourcecode, this is done in the subroutine `'count_pulses'`.

3. Decide which prescaler and which measuring interval should be used, depending on the coarse frequency measurement from step 2.

4. Reprogram the counter's prescaler so the divided input frequency is below 1 MHz (which is the maximum input frequency for the hardware counter, if the PIC is clocked with 4 MHz).
   If the coarse measured frequency is way below 1 MHz, the prescaler is turned off to get the best possible frequency resolution.

5. Count the pulses during the measuring interval (alias gate time), which is 0.25, 0.5, or 1 second. During this time, the display multiplexer keeps running. Overflows of the 8-bit timer register ("hardware") are counted by software in two other 8-bit registers, so the effective pulse counter has 24 bits (8 hardware bits plus 16 software bits while counting).

6. Gate time finished -> stop counting pulses.

7. If the hardware prescaler was active while counting (see step 4), multiply the pulse count with the prescaler ratio so we don't have to care for the prescaler setting in the following steps.
   If you know a bit about assembler programming: The multiplicator is always a power of two, so instead of a multiplication, the pulse count value (now expanded to 32 bit) is shifted left which is much easier on a PIC.

8. If the gate time was 0.5 seconds, multiply the pulse count by 2; if the gate time was 0.25 seconds, multiply the pulse count by 4. The result is the input frequency in Hertz, no matter which prescaler ratio or gate time was used. Like in the previous step, this "multiplication" is in fact a simple bit-shifting operation.

9. (Optional) Add the programmed frequency offset. If the result is negative, make it positive.

10. Split the frequency into eight (!) decimal digits. This is tricky with a PIC, see sourcecode. It is realized by repeatedly subtracting powers of ten from the 32-bit frequency value, beginning with ten millions (because the highest, theoretically possible frequency is 64 MHz).

11. Skip leading zeroes, and insert a decimal point after the kHz- or MHz digit (the kHz-point is ANDed with a blink flag)

12. Beginning with the first non-zero digit, convert five digits from binary code into seven-segment-patterns, and copy the result into the "display registers". The display multiplex routine which is executed while counting will write these registers to the LED display in steps 2 and 5 of the next main loop.

13. Poll the 'programming function' input ("RA5"). If this digital input is low, enter programming mode (not explained here). If not, go to step 1 to begin the next measurement.

Sounds tricky ? Well, you don't have to understand the internal function as long as you don't want to modify the firmware !

Links and other people's frequency counters

The DL-QRP AG use this counter in their "Super Dipper" (an interesting principle by the way). A description of the dipper -which contains the counter and preamp schematics- is available online here. The frequency counter is on a separate PCB, which is available as a kit from the DL-QRP-AG, see www.qrpproject.de/UK/DL4YHFcounter.html . Meanwhile, a second version with smaller display board is available; ideally suited for small monoband QRP transceivers.

My own counter prototype now does a nice job in Miss Mosquita (a neat little QRP transceiver). A short description of the counter in german language is here .

In 2011, Krzysztof (SQ3NQJ), made this variant of the counter firmware. It can be used with an external prescaler (divide by a power of two), to allow frequency measurements up to several hundred MHz (as much as the external prescaler allows). The external prescaler ratio can be configured in a new entry in the counter's setup menu. Thanks Krzysztof - several users had asked for such a feature !

Don, KC7ZOW, has more detailed construction notes, and screenshots of the config mode which may help you troubleshooting the display.

Erich, VK5HSE, added support for other crystal oscillator frequencies in his firmware variant. In July 2015, his modifications could be downloaded from github.com/erichVK5/DL4YHF-FrequencyCounterVK5Mods, along with Gerber data for a printed circuit board.

Lutz (DK3WI) built a portable counter which he describes here (in german language) .

Richard (VA3NDO) has some photos of his counter on his website. Like all the others mentioned below, his board looks much cleaner than my prototype !

V. Kocka-Amort made a variant with a divide-by-four prescaler, using 74AC74 high-speed flip-flops, Schmitt-Trigger inputs, and a preamplifier with BFS17A. The circuit diagram, PCB (for Eagle Version 5.9.0 'free edition'), modified firmware, and photos are in this zipped archive. It was tested up to 172 MHz, but may possibly work up to 200 MHz.

Lukasz (SQ2DYL) made a nice PCB for the counter, and translated the description into Polish language. His project can be found on this website of the SP-QRP group .

Renato (PY2RLM) made his own PCB of the keyer, which is a bit larger than mine but has everything on the board (not using the resistors as 'wires' between CPU and the display. The link to his website is here.

Jose Renato (PU2VFW) pointed me to this site with an article in portugese language. It contains a number of photos of the PCB, and the assembed kit.

Luciano, PY2BBS, put up his site in 2010 with updated plans - also in portugese language, including Renaud's 6-digit variant (see below).

Rahul (VU3WJM) created a new, single-sided PCB for the counter with 5 digits. He kindly made his layout available: You can download the copper tracks, the component overlay (with component values), and the solder stop mask as PDF files from here. (Note: the transistor drives the first digit, not the last one; but I don't have an up-to-date design).

Renaud (F8FII) made a modified version of the counter which drives 6 digits, for the expense of the auto-ranging function (it uses the former decimal point output to drive the extra digit; the decimal points on Renaud's display are hard-wired). Description of the modified hardware, and the modified firmware are on F8FII's website.

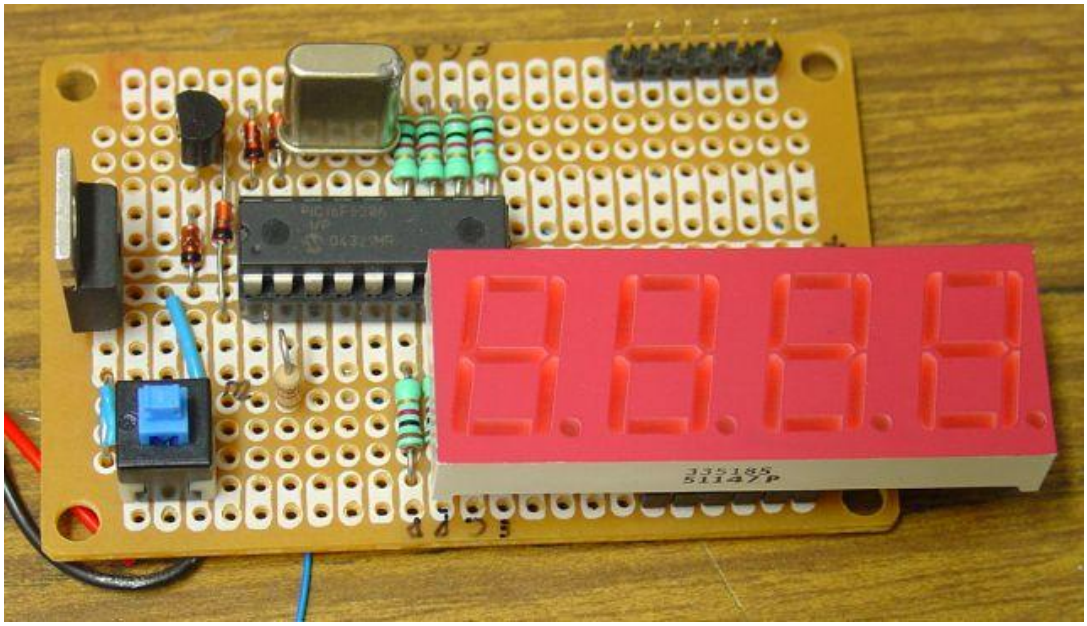
Jaak built this counter, with a more sensitive input amplifier using an NE592 .

Jan Panteltjie modified the counter firmware to send the measured frequency through the RS-232 port. Details are on Jan's website.

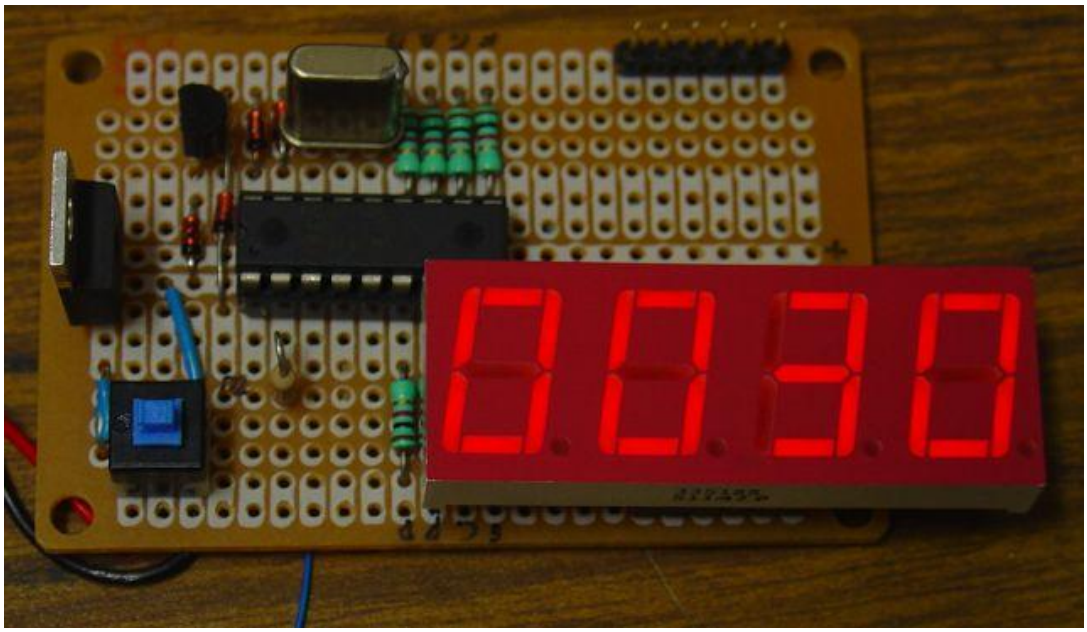
Joe (K3JLS) built a counter for his Century 21 transceiver - see pictures and description on his website .

Shig (JA1XRQ) translated the manual, and the construction details, into Japanese. The translated document is available in PDF format on his website.

Chetan (KG6NFG) built the counter with a single 4-digit module. Here are two pictures of his counter :



... and here how it glows in the dark :o)



Finally, there is another 'successful builder' who now sells an almost 1:1 reproduction of the 'DL4YHF counter' on ebay (with the minor addition of a crystal test oscillator). I don't mind doing that (since he sells the counter for a fair price, at least in February 2016), but unfortunately the kit maker / seller forgot to leave any note about the original developer and source. If you found *this site* via search engine, looking for "1Hz-50MHz Digital LED DIY Kits Crystal Oscillator Frequency Counter Tester", you will know the rest of the story. If you have problems with the kit mentioned above, please don't ask me to help you out, if there is something wrong with the counter (or the firmware, which seems to have been programmed erratically in a few occasions, with non-functional data EEPROM).

Back to the main index