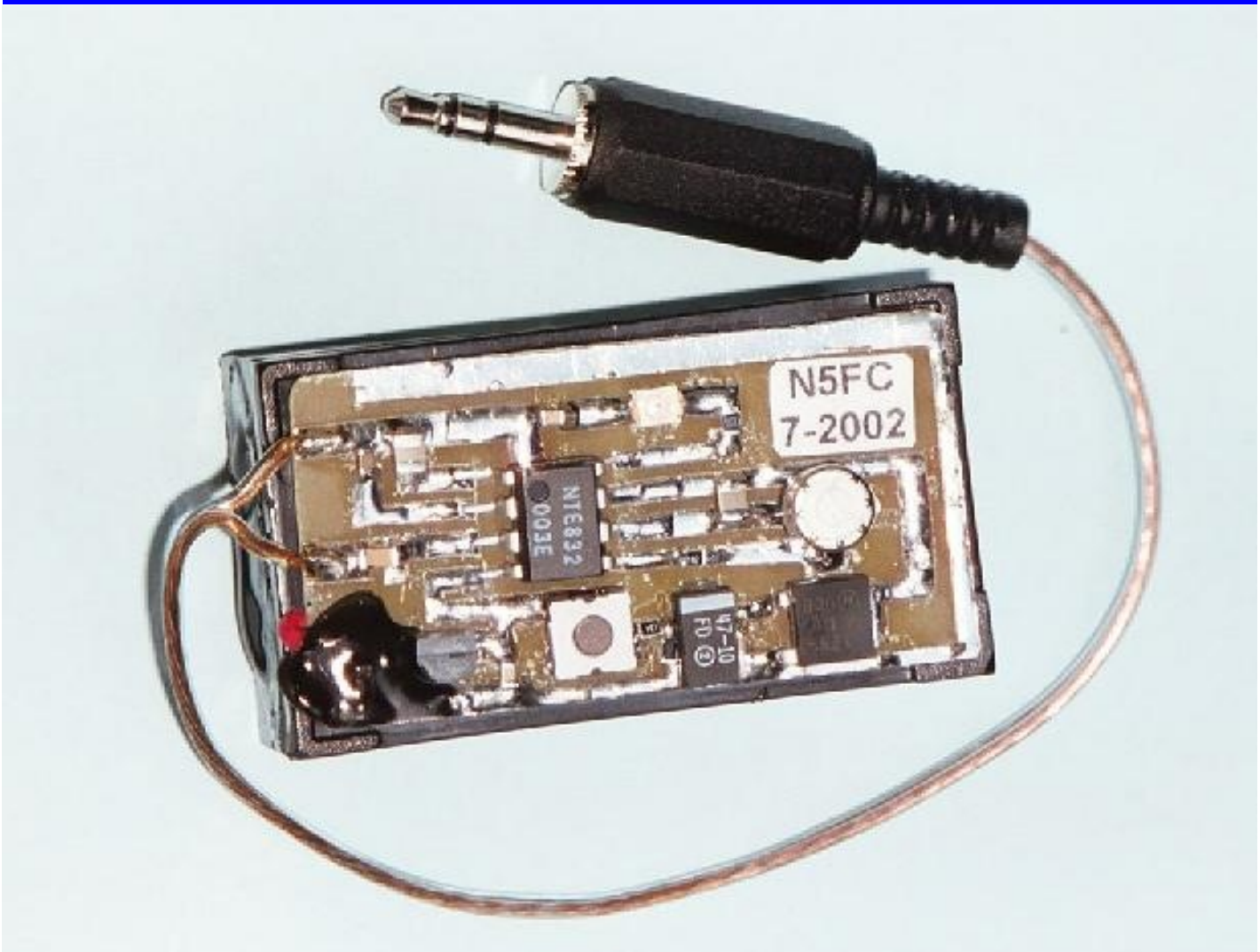
RAZzies

Maandblad van de Radio Amateurs Zoetermeer



Augustus 2022

Met in dit nummer:

- Micro SDR op een Pi-Pico (B) deel 1
- Opa Vonk: Tunen van resonante antennes
- WRTC contest
- Zero-beat afstemhulp
- PA3CNO's Blog

Colofon

RAZzies is een uitgave van de Radio Amateurs Zoetermeer. Bijeenkomsten van de Radio Amateurs Zoetermeer vinden plaats op elke tweede en vierde woensdag van de maanden september - juni om 20:00 uur in het clubhuis van de Midgetgolfclub Zoetermeer in het Vernède sportpark in Zoetermeer.

Website:

http://www.pi4raz.nl

Redactie:

Frank Waarsenburg PA3CNO pa3cno@pi4raz.nl

Eindredactie:

Robert de Kok PA2RDK pa2rdk@pi4raz.nl

Informatie:

info@pi4raz.nl

Kopij en op- of aanmerkingen kunnen verstuurd worden naar razzies@pi4raz.nl

Nieuwsbrief:

http://pi4raz.nl/maillist/subscribe.php

Van de redactie

Deze uitgave ben ik van mijn principes afgeweken. Dat wil zeggen: ik heb een verschrikkelijke hekel aan vervolgverhalen, of het nou op schrift of op de TV is. Zit je twee uur te kijken naar iets, verschijnt er ineens "to be continued" in beeld. Maar dat vervolg ga ik dan nooit zien. Vergeten, andere dingen te doen, je kent dat wel. Ik heb dan ook nog nooit een soap gezien (nog afgezien van waarom je je hersens aan zoiets bloot zou willen stellen). Maar deze keer was het voor mij onvermijdelijk. Het artikel van Arjan PE1ATM was dusdanig groot dat het anders meer dan 60 pagina's zou beslaan, ook al omdat ik de Engelse brontekst ongewijzigd heb overgenomen waardoor de benodigde ruimte natuurlijk twee keer zo groot is. Dus is het artikel opgesplitst in twee delen, waarvan het eerste deel in deze uitgave te lezen is. En nog kom ik dan aan de 37 pagina's. Maar ik ben er maar wat blij mee, want al meer dan 10 jaar elke maand een blad vol krijgen begint best een uitdaging te worden. Daardoor heb ik ook voor het eerst in de historie van de RAZzies een artikel een maand door moeten schuiven. Een luxeprobleem zullen we maar zeggen. Alleen moeten jullie een maandje wachten op het vervolg. Verder: steek je set eens aan! De condities zijn de laatste tijd best goed. De hoge banden zijn steeds vaker goed open en ik krijg het zelfs voor elkaar om op 6m verbindingen te maken met slechts een Inverted-V!

Micro SDR op een Pi-Pico (1)

Arjan te Marvelde, PE1ATM

Sinds 2020 biedt Raspberry een nieuwe module aan op basis van hun eigen ontwikkelde processor, de **RP2040**. Deze processor bevat een dual core, 125MHz Cortex-gebaseerde controller met veel Flash en RAM. Hij heeft veel zeer configureerbare I/O-pinnen, waardoor de toepassing van deze module een fluitje van een cent is. De C-SDK lijkt nog niet volledig volwassen te zijn (in mei 2021) maar geeft in ieder geval een snelle start om deze **Pi-Pico** daadwerkelijk in gebruik te nemen.

Dit artikel beschrijft een testimplementatie van een kleine SDR, geïnspireerd op de QCX van Hans

Micro SDR on a Pi-Pico (1)

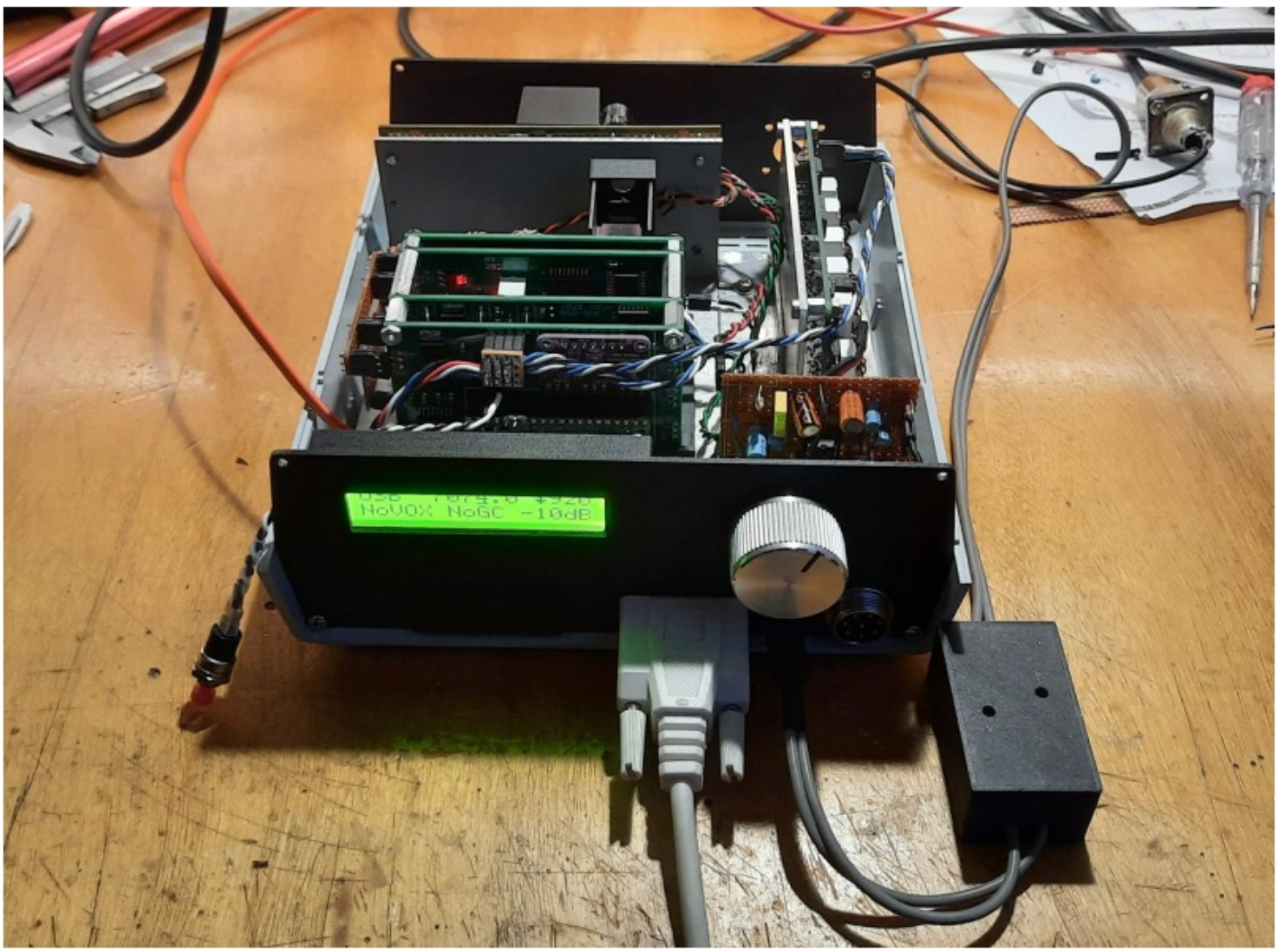
Arjan te Marvelde, PE1ATM

Since 2020, Raspberry offers a new module based on their own developed processor, the **RP2040**. This processor contains a dual core, 125MHz Cortex based controller with plenty of Flash and RAM. It has many highly configurable I/O pins, which makes application of this module a breeze. The C-SDK does not seem to be fully mature yet (in May 2021) but at least it provides a quick start in actually setting this **Pi-Pico** to use.

This article describes a test implementation of a small SDR, inspired by Hans Summers' QCX and

Summers en alles wat daarna volgde, zoals μSDX. Een belangrijke afwijking in hardware is het gebruik van op FST3253 gebaseerde mixers voor zowel RX- als TX-paden, wat modularisatie en experimenteren een beetje eenvoudiger maakt. De RX en TX front-ends, de besturing, signaalverwerking, audio i/f en mixer onderdelen kunnen in principe apart gebouwd worden. Zie het blokschema.

everything that followed after that, such as μSDX. A main deviation in hardware is the use of FST3253 based mixers for both RX and TX paths, which makes modularization and experimentation a bit easier. The RX and TX front-ends, the control, signalprocessing, audio interface and mixer parts can in principle be built separately. Refer to the block schematic.



De afbeelding hierboven toont de testopstelling op het moment van schrijven, waarbij de RX en TX onderdelen werken en modules zijn ingebouwd in een Teko-behuizing. Sinds v2.02 zijn verschillende bugs opgelost en is de code geherstructureerd om het invoegen van de nieuwe FFT-gebaseerde signaalverwerking in het frequentiedomein mogelijk te maken.

The image above shows the test setup at the date of writing, where the RX an TX parts are working and modules are built into a Teko enclosure. Since v2.02 several bugs have been resolved, and the code has been restructured to allow insertion of the new FFT-based frequency domain signal processing.

Opmerking voor bouwers: dit project is zeer experimenteel en zal een werk in uitvoering blijven, ik probeer er zo nu en dan wat tijd aan te

Note to builders: This project is highly experimental and will remain a work in progress, I try to spend some time on it every now and

besteden, maar het is gewoon een hobby. Dus voel je vrij om dit project te kopiëren en toe te voegen of te veranderen wat je maar wilt. Het is bedoeld om mee te experimenteren en mag daarom niet worden beschouwd als een perfect werkende kit.

then but it is just hobby. So, feel free to copy this project and add or change whatever you like. It is intended for experimentation and hence should not be considered as a flawlessly working kit.

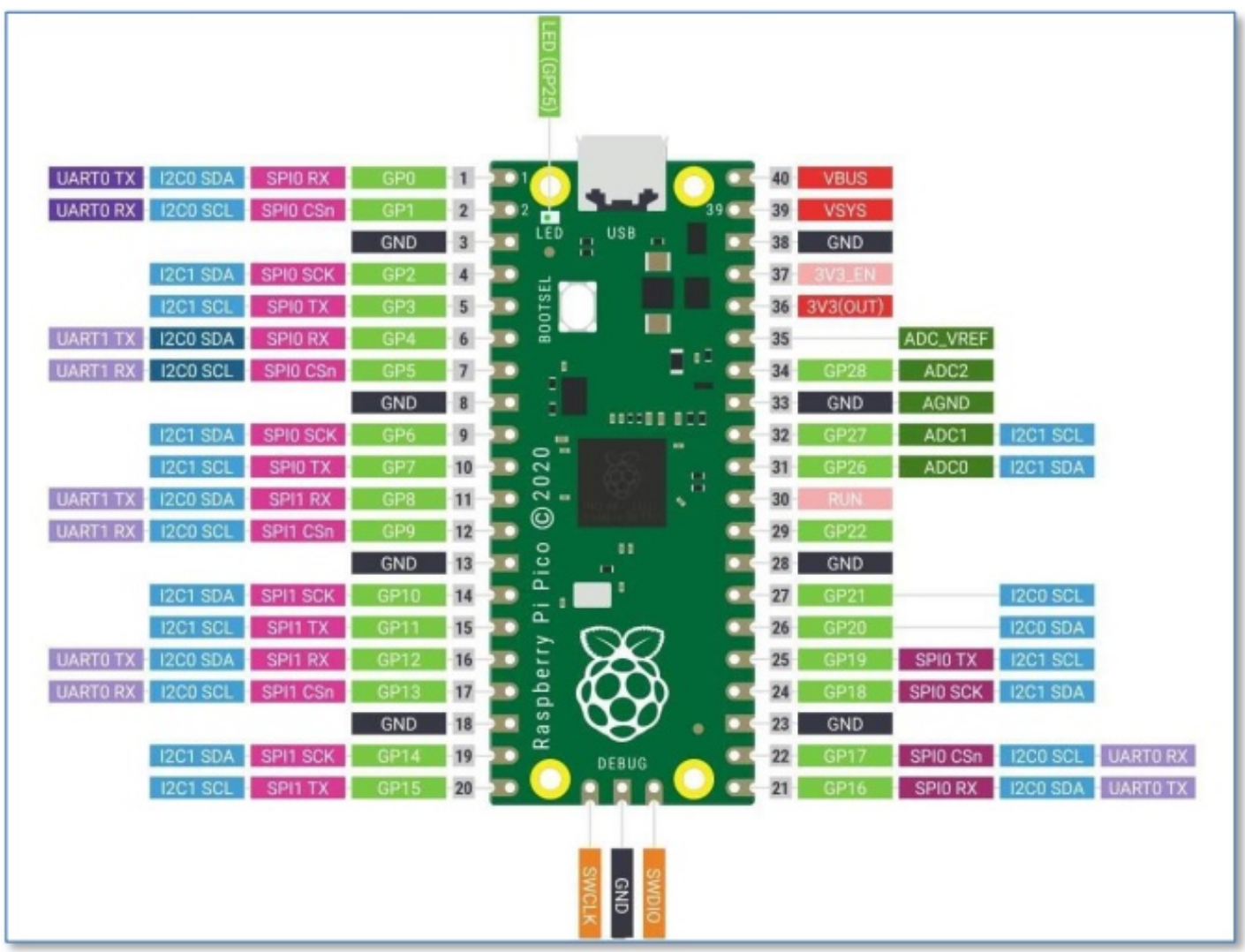
Het project wordt onderhouden op GitHUB: https://github.com/ArjanteMarvelde/uSDR-pico. Deze versie is getest en klaar om als startpunt voor je SDR-experimenten te worden gebruikt.

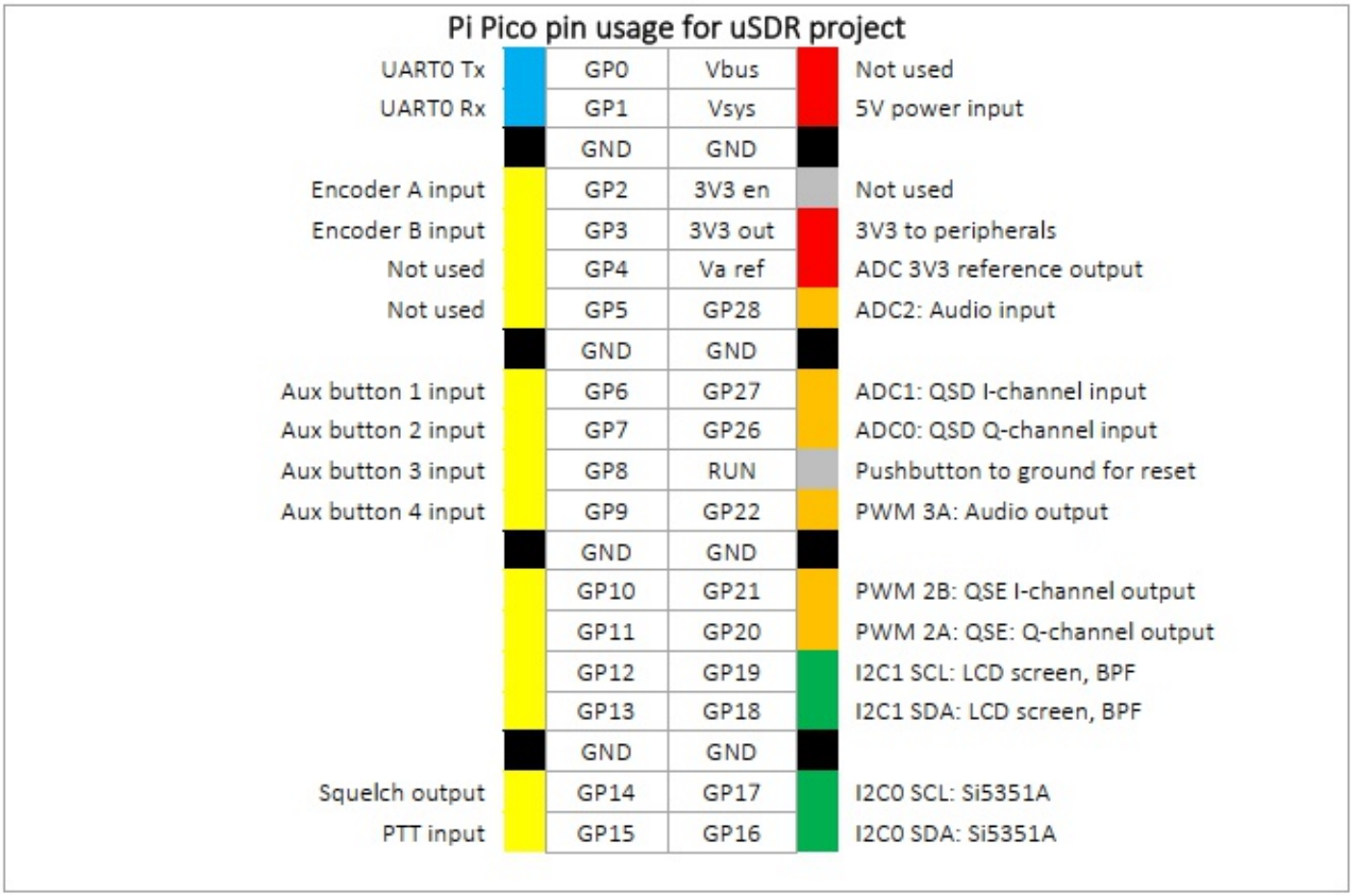
The project is maintained on GitHUB: https://github.com/ArjanteMarvelde/uSDR-pico. This version has been tested, and is ready to be used as starting point for your SDR experiments.

1 Raspberry Pi Pico (RP2040)

De Pi-Pico-module vormt het hart van de implementatie, dus hier is de pinout en de manier waarop deze in dit project wordt gebruikt.

The Pi-Pico module is the heart of the implementation, so here is the pinout and the way it is used in this project.





2 Werkingsprincipe

Het blokschema op de volgende pagina toont het processor board en verschillende rand-apparatuur. De VFO is gebaseerd op een Si5351, dit kan bijvoorbeeld een Adafruit-bord zijn. De VFO klokt de ontvanger (Quadrature Sampling Detector, QSD) en zender (Quadrature Sampling Exciter, QSE) mixers, die zijn gebaseerd op de FST3253. In principe kan alles worden gebruikt dat een kwadratuur RX-signaal produceert en een kwadratuur TX-signaal accepteert. Er zijn 4 GPIO's gereser-veerd voor het schakelen van bandfilters.

Houd er rekening mee dat de ADC's (en DAC'sO) een maximaal bereik van 3V3 hebben en dienovereenkomstig moeten worden beperkt.

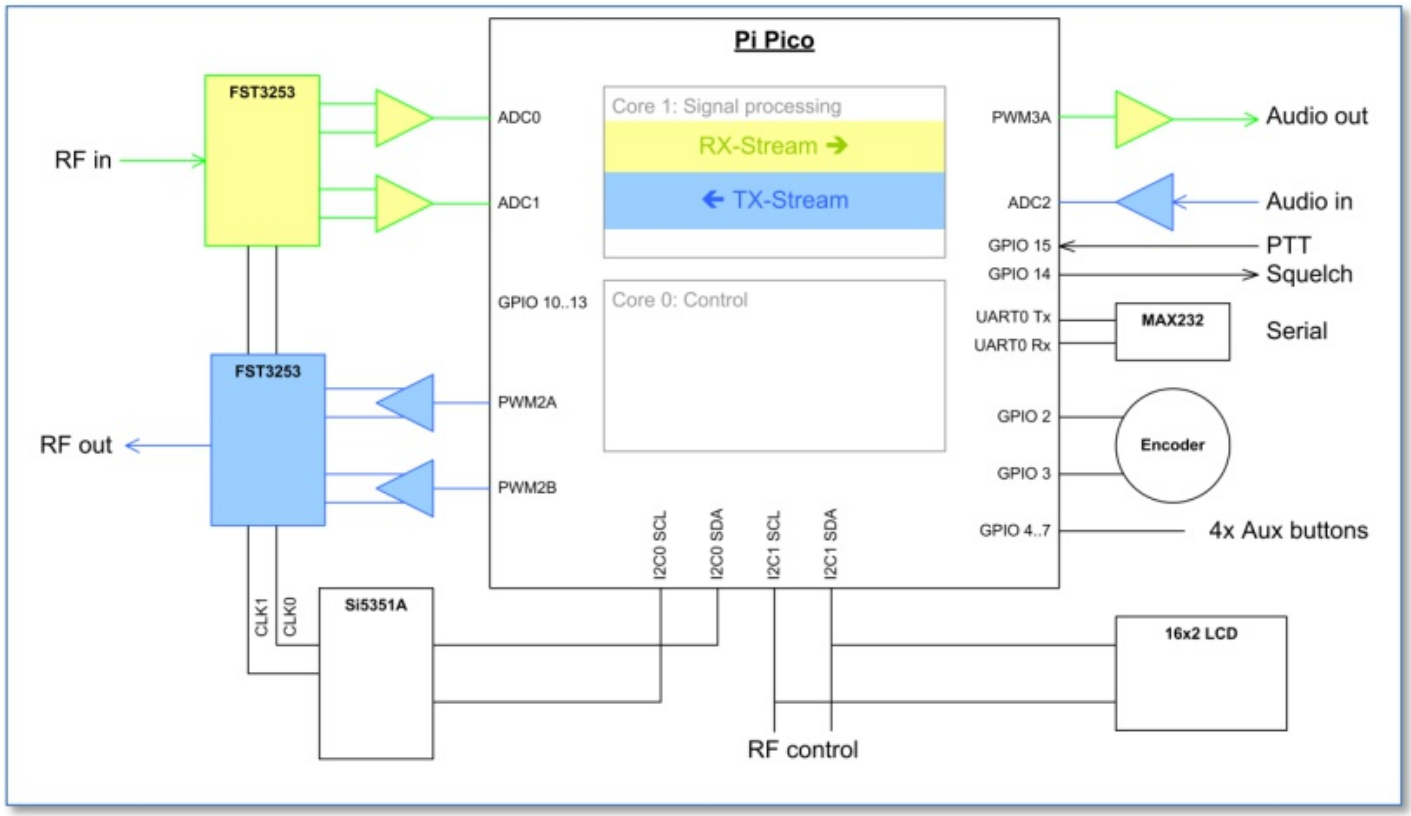
Aan de gebruikerskant zijn er interfaces voor audio, PTT/Squelch, een rotary encoder, 4 (tot 8) hulpknoppen, een seriële poort voor een pc-

2 Principle of operation

The block diagram on the next page shows the processor board and several peripherals. The VFO is based on a Si5351, which may be for example an Adafruit board. The VFO clocks the receiver (Quadrature Sampling Detector, QSD) and transmitter (Quadrature Sampling Exciter, QSE) mixers, which are based on the FST3253. In principle anything can be used that produces a quadrature RX signal and consumes a quadrature TX signal. There are 4 GPIOs reserved for band filter switching.

Note that the ADCs (and DACsO have a maximum range of 3V3 and should be limited accordingly.

On the user side, there are interfaces for Audio, PTT/Squelch, a Rotary Encoder, 4 (up to 8) Auxiliary Buttons, a serial port for a PC interface



interface en een andere I²C-interface om zowel het LCD-scherm als het HF-front-end te bedienen (BPF, LNA, demping).

and another I²C interface to control LCD as well as the RF front end (BPF, LNA, Attenuation).

Natuurlijk is er de USB-interface die wordt gebruikt voor het programmeren van apparaten, en deze kan ook worden gebruikt als op stdio gebaseerde debug-monitor in plaats van de UART op pinnen 1 en 2.

Of course there is the USB interface which is used for device programming, and this could also be used as stdio based debug monitor instead of the UART on pins 1 and 2.

3 Software

Samenvatting:

De processor heeft twee cores die in grote mate onafhankelijk werken, afgezien van wat onregelmatig wachten veroorzaakt door arbitrage bij geheugentoegang. Het idee is om core1 alle signaalverwerking te laten doen, terwijl core0 (de standaardinstelling bij het opstarten) alle controles doet.

Summary:

The processor has two cores that work to a large extent independently, apart from some hit and miss waits caused by memory access arbitration. The idea is to let core1 do all the signal processing, while core0 (the default at startup time) does all the control stuff.

Merk op dat de Pico een eXecute In Place (XIP) Flash-interface gebruikt, wat betekent dat de code echt wordt uitgevoerd vanuit een relatief klein cachegeheugengedeelte van het SRAM. Normaal gesproken is dit oké, maar tijdkritische

Note that the Pico uses an eXecute In Place (XIP) Flash interface, meaning that code is really executed from a relatively small cache-memory section of the SRAM. Normally this is okay, but time-critical parts of the code can also be loaded

delen van de code kunnen ook tijdens het opstarten in SRAM worden geladen; de Pico heeft genoeg geheugen. Ook bevinden de tijdkritische zaken zich in core1, en daarom krijgt deze core prioriteit boven core0 in geval van arbitrage voor toegang tot gedeelde bronnen.

in SRAM at boot time; the Pico has plenty memory. Also, the time critical stuff is located in core1, and therefore this core is given priority over core0 in case of arbitration for access to shared resources.

De signaalverwerking in core1 is opgesplitst in twee streams, een RX- en een TX-stream. Alle drie de ADC-ingangen worden gesampled op de hoogste snelheid in Round-Robin-mode, en de interrupt-handler kopieert alleen de meest recente conversieresultaten naar buffers wanneer de ADC FIFO vol raakt. Op deze manier worden ADC-samples voor elk kanaal genomen gedurende een periode van 6 µsec ergens binnen het bemonsteringsritme.

The signal processing on core1 is split up in two streams, an RX and a TX stream. All three ADC inputs are sampled on highest speed in Round-Robin fashion, and the interrupt handler merely copies the most recent conversion results into buffers when the ADC FIFO fills up. This way, ADC samples for every channel are taken during a period of 6 µsec somewhere within the sampling rythm.

De RX-stream neemt de I- en Q-samples van de QSD, verwerkt deze en voert samples uit via een PWM-gebaseerde DAC naar de audio-interface. De TX-stream doet het omgekeerde, neemt de samples van de audio-ingang, verwerkt ze en stuurt PWM-DAC I- en Q-signalen naar de QSE.

The RX stream takes the I and Q samples from the QSD, processes these and outputs samples through a PWM-based DAC to the audio interface. The TX stream does the reverse, taking the samples from the audio input, process them and sending PWM-DAC I and Q signals to the QSE.

Voor de verwerking van de samples kan tijdens het compileren een van de twee mechanismen worden geselecteerd; de eerste verwerkt alles in het tijdsdomein terwijl de tweede verwerkt in het frequentiedomein. Het tijddomeinmechanisme verwerkt het signaal monster voor monster en heeft daarom een zeer korte loop. Het frequentiedomeinmechanisme verzamelt samples in een buffer die door middel van een FFT wordt omgezet naar het frequentiedomein. Na verwerking gebeurt het omgekeerde met een iFFT.

For the processing of the samples, one out of two mechanisms can be selected during compile time; the first processes everything in the time domain while the second processes in the frequency domain. The time-domain mechanism processes the signal sample-by-sample, and therefore has a very short loop. The frequency domain mechanism collects samples in a buffer which is converted to frequency domain by means of an FFT. After processing the reverse is done with an iFFT.

De verwerking van het tijdsdomein wordt uitgevoerd in periodes van 64 sec (15,625 kHz) en de verwerking van het frequentiedomein elke 32,768 msec (d.w.z. 512 x 6 sec). De core1 timer callback-functie zorgt voor de sample-overdracht en activeert de DSP-loop op het juiste moment. De DSP-hoofdloop verwerkt de eigenlijke RX- en TX-streams.

The time domain processing runs on a 64µsec basis (15.625 kHz) and the frequency domain processing every 32.768msec (i.e. 512 x 6 µsec). The core1 timer callback function takes care of the sample transfer and triggers the DSP loop at the right moment. The DSP main loop processes the actual RX and TX streams.

De Pico heeft twee I^2C-bussen, dus we kunnen

The Pico has two I^2C buses, so we can make it

het gemakkelijk maken en een snelle bus aansluiten aan de Si5351 en de andere gebruiken om de overige apparaten te bedienen. Er zijn vier GPIO's gereserveerd voor hulpknoppen. Deze worden momenteel gebruikt als directe lijnen, maar zouden gemakkelijk met een binaire decoder kunnen worden uitgebreid om het aantal controlelijnen te vergroten.

easy and connect a fast one dedicated to the Si5351 and use the other to control the remaining devices. There are four GPIOs reserved for auxiliary buttons. These are currently used as direct controls, but this could easily be upgraded with a binary decoder to increase the number of control lines.

Bestandsoverzicht:

uSDR.c De hoofdloop en systeeminitialisatie.
De hoofdloop draait als een afzonderlijk proces op core0 en zorgt voor alle gebruikersinterfaces.

dsp.c Signaalverwerkingsloop, verwerkt de bemonstering en roept TX- en RX-routines aan.
Deze loop wordt uitgevoerd als een afzonderlijk proces op core1.

dsp_tim.c Tijddomein signaalverwerkingsengine

dsp_fft.c Frequentiedomein signaalverwerkingsengine

fix_fft.c Fixed-point Fast Fourier Transformatie functies

si5351.c Besturing van de VFO-module, die I/Q-klokken levert aan de QSD en QSE.

lcd.c LCD-uitvoerondersteuning en 16x2 byte uitvoerbuffer.

hmi.c De gebruikersinteractie, het afhandelen van de besturingsevents.

monitor.c Een opdrachtshell die draait op de stdio UART.

relay.c Bestuurt de diverse relais via de I2C uitbreidingen.

Files overview:

uSDR.c The main loop and system initialization.
The main loop runs as a separate process on core0 and takes care of all user interfacing.

dsp.c Signal processing loop, handles the sampling and invokes TX and RX branches.
This loop runs as a separate process on core1.

dsp_tim.c Time domain signal processing engine

dsp_fft.c Frequency domain signal processing engine

fix_fft.c Fixed-point Fast Fourier Transform functions

si5351.c Control of the VFO module, that provides I/Q clocks to the QSD and QSE.

lcd.c LCD output support and 16x2 byte output buffer.

hmi.c The user interaction, handling the control events.

monitor.c A command shell running on the stdio UART.

relay.c Controls the various relays through I2C expanders.

3.1 Signaalverwerking

Waar alle andere onderdelen functies activeren of besturen, vormt de signaalverwerking het hart van de uSDR-Pico. Dit is waar dit project echt over gaat.

3.1.1 Sampling en timing (dsp.c)

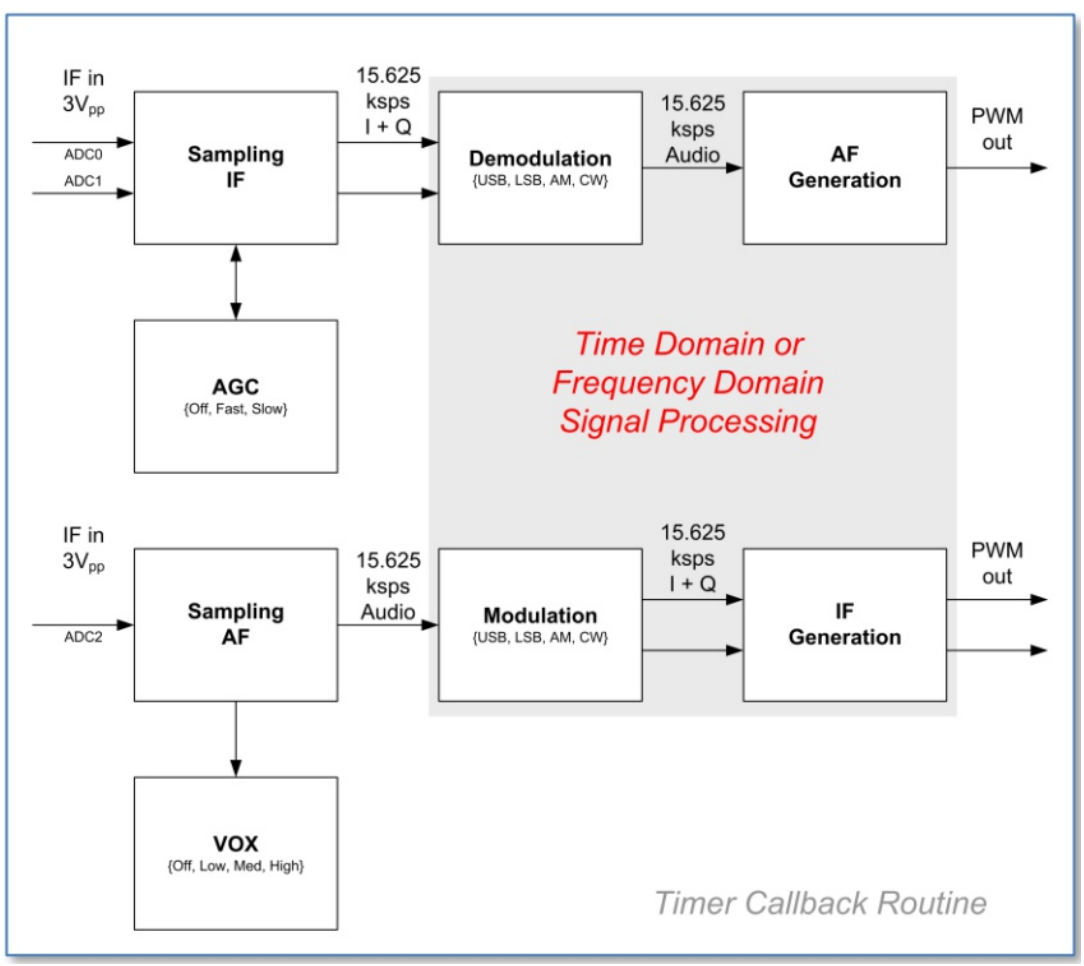
De structuur is zo gebouwd dat de signaalverwerking in het tijd- of frequentiedomein tijdens het compileren kan worden geselecteerd door een constante in dsp.h te definiëren. Het hoofdgedeelte in dsp.c bevat de functies om de samples te verwerven, om wat voorbewerkingen uit te voeren zoals niveaudetectie t.b.v. AGC en VOX en om verder te zorgen voor de timing. Deze timing werkt natuurlijk bij beide methodes anders; in het geval van het tijddomein (TD) moet de RX-stroom worden aangeroepen voor elke sample en in het geval van het frequentiedomein (FD) is dit alleen elke keer dat een FFT-buffer vol is.

3.1 Signal Processing

Where all other parts are enabling or control functions, the signal processing forms the heart of the uSDR-Pico. This is what this project is really about.

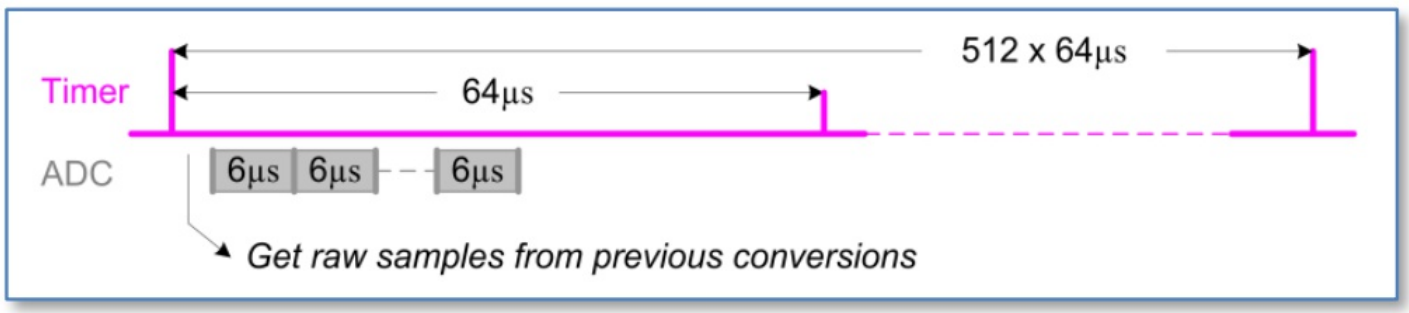
3.1.1 Sampling and timing (dsp.c)

The structure is built in such a way, that the time or frequency domain signal processing can be selected at compile time by defining a constant in dsp.h. The main part in dsp.c contains the functions to acquire the samples, to do some preprocessing like level detection for AGC and VOX and further take care of the timing. This timing obviously works different in both methods; in the time-domain (TD) case the RX stream needs to be invoked for each sample and in the frequency domain (FD) case this is only every time an FFT buffer is filled.



Het bemonsteringsproces gebruikt de ADC's op free-running round robin-manier voor alle drie de kanalen, waarbij de monsters worden opgeslagen in speciale buffers in de ADC-FIFO IRQ-routine. Elke conversie duurt 2µsec, na 3 conversies genereert de ADC FIFO een IRQ en stopt de ISR de conversies.

The sampling process uses the ADCs in free-running round robin fashion for all three channels, storing the samples in dedicated buffers in the ADC-FIFO IRQ routine. Each conversion takes 2µsec, after 3 conversions the ADC FIFO generates an IRQ and the ISR stops the conversions.



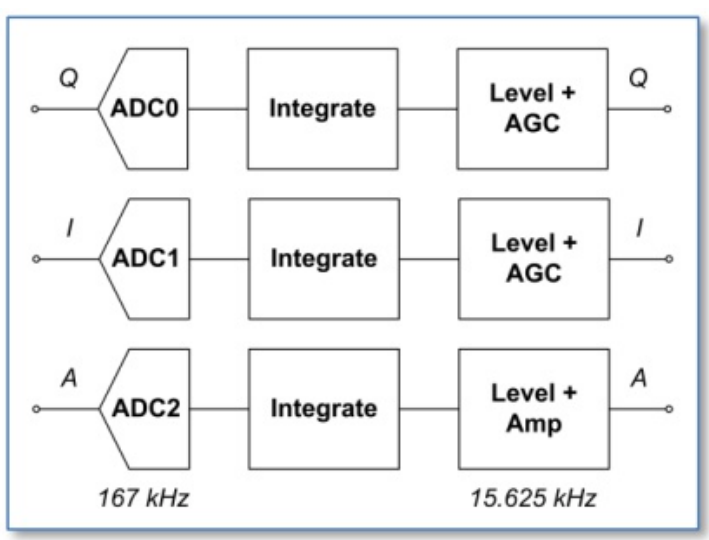
De afbeelding hierboven toont de timing van de ADC-FIFO en Timer IRQ callback-routines. De laatste samples worden uit de buffers gehaald aan het begin van elk 64µsec tijdslot, voordat een nieuwe ADC-conversiecyclus wordt gestart. Deze cyclus kan tot 10 samples per kanaal integreren (d.w.z. 10x 6µsec) voor een groter dynamisch bereik.

The image above shows the timing of the ADC-FIFO and Timer IRQ callback routines. The latest samples are taken from the buffers at the beginning of each 64µsec time slot, before starting a new ADC conversion cycle. This cycle could integrate up to 10 samples per channel (i.e. 10x 6µsec) for increased dynamic range.

Voorbewerking van monsters wordt elk tijdslot in de timer-callback-routine gedaan, maar zowel tijd- als frequentiedomeinverwerking wordt uitgevoerd als een achtergrondproces, dat wordt geactiveerd na de voorbewerking. De verwerking van het tijddomein wordt elke slot geactiveerd, terwijl de verwerking van het frequentiedomein pas om de 512 slots wordt gestart wanneer een ½ FFT-buffer is gevuld.

Sample preprocessing is done every timeslot in the timer callback routine, but both time and frequency domain processing is executed as a background process, which is triggered after the preprocessing. Time domain processing is triggered every slot, while frequency domain processing is only started every 512 slots when a ½ FFT buffer is filled.

3.1.1.1 Sample preprocessing



De timer-callback-routine bepaalt dus de werkelijke bemonsteringsfrequentie die is ingesteld op 64 µsec (15,625 kHz).

De fasefout tussen de I- en Q-samples is de duur van één ADC-conversie; 2 µsec. Dit resulteert in een faseverschil van ongeveer 1% in het audiodomein (bij een frequentie van 4 kHz), en daarom is er geen echte noodzaak voor intermitterende bemonstering en de vereiste fasecorrectie door het middelen van de laatste twee monsters op het I-kanaal. Een dergelijk middelingsproces zou in feite resulteren in een veel grotere vervorming.

DC-verwijdering

De samples van de ADC zijn elektrisch gecentreerd op ongeveer de helft van de referentiespanning, wat de helft is van het dynamische bereik van de ADC (2048). Dit niveau wordt afgetrokken voor het integreren in de ADC-FIFO-callback-routine.

Een nauwkeuriger DC-verwijderingsproces wordt nog steeds aanbevolen, om het laagdoorlaatgefilterde lopende gemiddelde signaal af te trekken. Dit kan echter te veel overhead kosten en kan worden weggelaten.

dc += (sample – dc)/128
De RC tijd is 127*64µsec ≈ 8msec of 125Hz 1e orde laagdoorlaat

Merk op dat in dit schema alleen monsters die groter zijn dan dc ± 128 daadwerkelijk zullen bijdragen aan een correctie, dus prescaling is vereist om dit te voorkomen.

AGC

Een schatting van het signaalniveau wordt bijgehouden door laagdoorlaatfiltering van de absolute waarde van de DC-gecorrigeerde samples. De I- en Q-stromen worden versterkt met de AGC-versterkingsfactor, die is afgeleid van het laagdoorlaatgefilterde ADC-niveau van deze I- en Q-kanalen. De AGC is slechts een

So, the timer callback routine determines the actual sampling rate which is set to 64 µsec (15.625 kHz).

The phase error between the I- and Q-samples is the duration of one ADC conversions; 2 µsec. This results in a phase difference of about 1% in the audio domain (at a frequency of 4kHz), and hence there is no real need for intermittent sampling and the required phase correction by averaging the last two samples on the I channel. Such an averaging process in fact would result in a much larger distortion.

DC removal

The samples from the ADC are electrically centered on approximately half the reference voltage, which is half the ADC dynamic range (2048). This level is subtracted before integration in the ADC-FIFO callback routine.

A more accurate DC removal process is still recommended, to subtract the low-pass filtered running average signal. This may however take too much overhead and could be omitted.

dc += (sample – dc)/128
The RC time is 127*64µsec ≈ 8msec or 125Hz 1st order low pass

Note that in this scheme only samples that are larger than dc ± 128 will actually contribute to a correction, so prescaling is required to prevent this.

AGC

A signal level estimate is maintained by low-pass filtering the absolute value of the DC-corrected samples. The I and Q streams are amplified with the AGC gain factor, which is derived from the low pass filtered ADC level of these I and Q channels. The AGC is just a multiplication factor for the samples, to stretch

vermenigvuldigingsfactor voor de monsters, om ze tot het gewenste bereik uit te rekken.

Merk op dat dit bereik verschilt voor TD- en FD-verwerking.

VOX

De spraakgestuurde schakelaar (VOX)-procedure moet het audio-ADC-niveau continu testen, dus dit wordt zowel in RX- als in TX-modus uitgevoerd. De VOX neemt het audiostreamniveau en voert het door een laagdoorlaatfilter. De VOX-hangtijd bepaalt hoe lang het actief blijft nadat het niveau onder de VOX-drempel is gedaald.

Het audioniveau wordt bepaald in de timer-callback-routine, maar de eigenlijke VOX-detector wordt geïmplementeerd in de DSP main loop.

3.1.1.2 Sample uitvoer

De uitvoer van samples naar de audio-interface in RX-mode of de QSE in TX-mode wordt ook afgehandeld door de timer-callback-routine, net voordat de RX- of TX-signaalverwerking wordt aangeroepen. Het signaal wordt op het bereik gecontroleerd en ofwel afgekapt of verzwakt voordat het naar de DAC wordt gestuurd.

3.1.1.3 Aanroepen van de signaalverwerking

Voor de TD-verwerking is de effectieve bemonsteringssnelheid ook de snelheid voor de RX/TX-signaalverwerkingsroutines. Voor de FD-verwerking worden de RX/TX-functies alleen aangeroepen om de halve FFT-grootte (1024/2) aan monsters, d.w.z. wanneer een buffer is gevuld. Dit levert een veel lagere aanroep-snelheid op van 305 Hz (1/32.768 msec), maar dit wordt uiteraard gecompenseerd door de grotere verwerkingsbelasting die wordt opgelegd door de Fourier-transformaties.

them to the desired range.

Note that this range is different for TD and FD processing.

VOX

The voice activated switch (VOX) procedure needs to continuously test the Audio ADC level, so this is performed both in RX and TX mode. The VOX takes the Audio stream level and takes it through a low-pass filter. The VOX linger time determines how long it will remain active after the level dropped below the VOX threshold.

The audio level is determined in the Timer callback routine, but the actual VOX detector is implemented in the DSP main loop.

3.1.1.2 Sample output

The output of samples to either the audio interface in RX mode or the QSE in TX mode is also handled by the timer callback routine, just before invoking the RX or TX signal processing. The signal is range checked and either clipped or attenuated before sending it to the DAC.

3.1.1.3 Invoking the signal processing

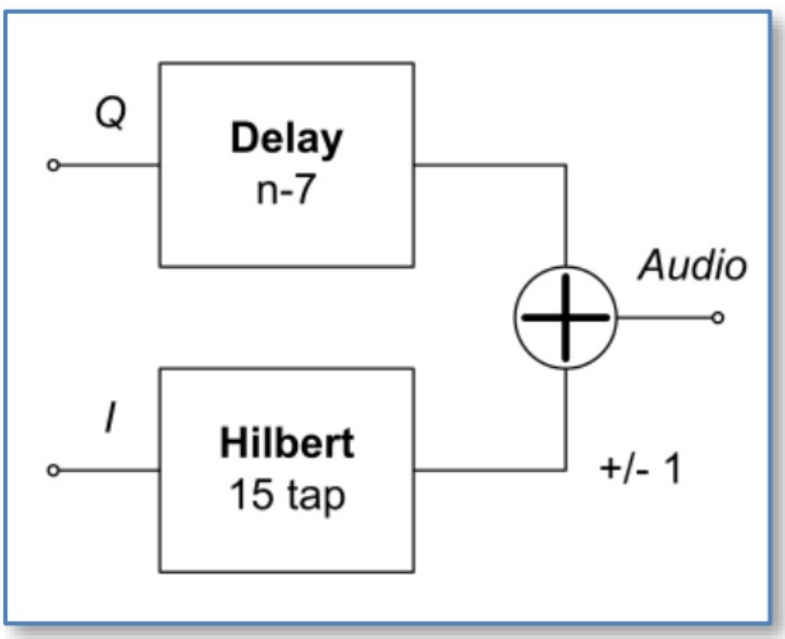
For the TD processing the effective sample rate is also the rate for the RX/TX signal processing routines. For the FD processing the RX/TX functions are invoked only every half FFT-size (1024/2) samples, i.e. when a buffer is filled. This yields a much lower call rate of 305Hz (1/32.768msec), but this obviously is compensated by the larger processing load imposed by the Fourier transformations.

3.1.2 Tijdsdomein signaalverwerking (dsp_tim.c)

3.1.2.1 RX-stream

De RX-stream kan functioneel worden gesegmenteerd in een deel dat het eigenlijke gekozen type demodulatie doet, en een deel dat de audiogeneratie doet.

Demodulatie



Voor SSB-demodulatie worden de binnenkomende I- en Q-samples opgeslagen in een vertragingslijn van 15 samples. Er wordt een 15-segments Hilbert-transformatie op het Q-kanaal uitgevoerd en het resultaat wordt afgetrokken van of opgeteld bij de n-7-sample in de I-vertragingslijn om respectievelijk de USB- of LSB-audio-uitvoer te verkrijgen.

Voor AM-demodulatie moet de lengte van de I-Q-vector worden berekend en is er geen verdere transformatie vereist.

De resulterende audiosamplestream wordt overgedragen aan het audiogeneratieproces.

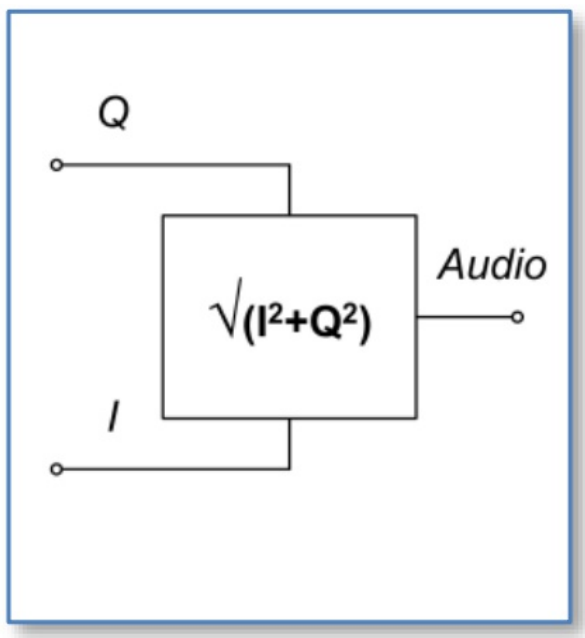
Zie de Hilbert-transformatie op de volgende bladzijde

3.1.2 Time domain signal processing (dsp_tim.c)

3.1.2.1 RX stream

The RX stream can be functionally segmented into a part that does the actual demodulation type of choice, and a part that does the audio generation.

Demodulation

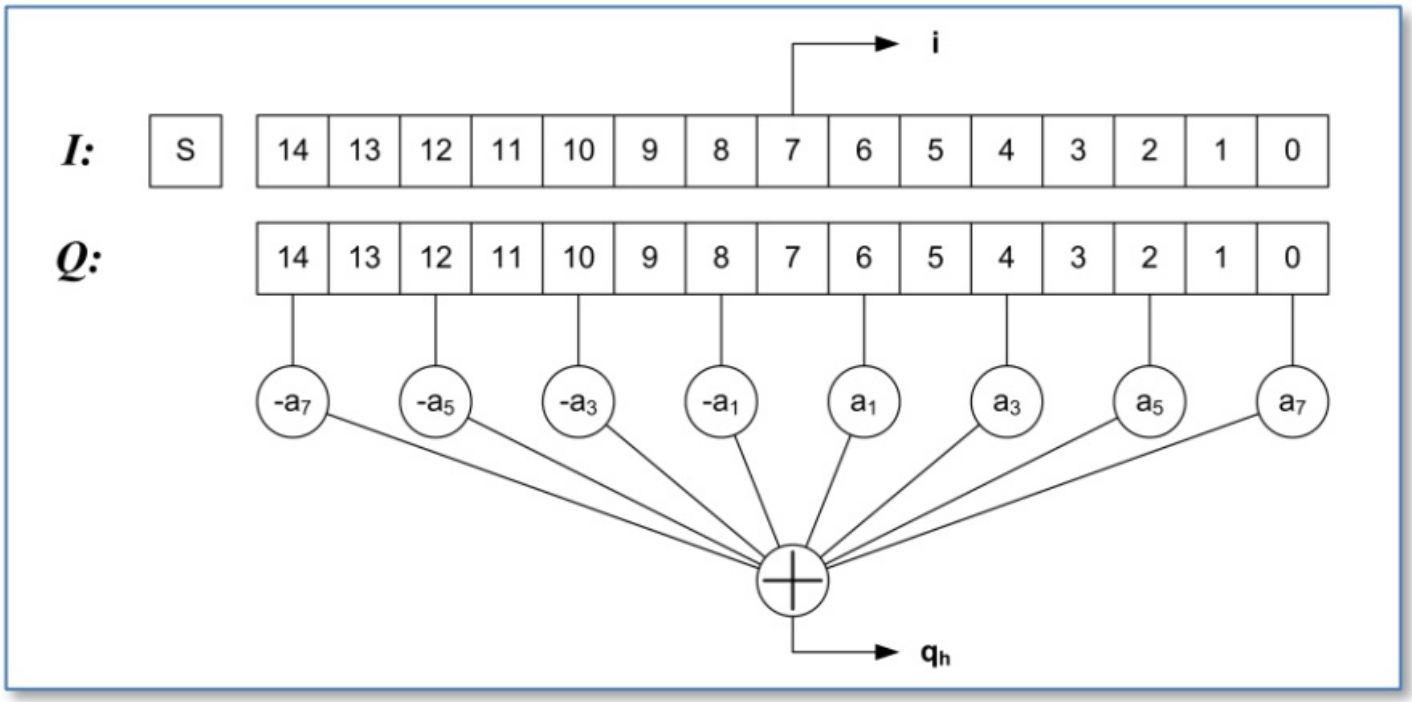


For SSB demodulation, the incoming I and Q samples are stored in a 15-sample delay line. A 15-tap Hilbert transform on the Q channel is done and the result is subtracted from or added to the n-7 sample in the I delay line to obtain the USB or LSB audio output respectively.

For AM demodulation the length of the I-Q vector needs to be calculated, and no further transform is required.

The resulting audio sample stream is handed to the Audio generation process.

See the Hilbert transform on the next page



De Q-delay line-array heeft een lengte van 15, om een klassieke Hilbert-transformatie met 15 segmenten mogelijk te maken. De even samples hebben een coëfficiënt van nul, dus zoals in de bovenstaande figuur is te zien worden slechts 8 van de samples gebruikt in de berekening. Vanwege de symmetrie van deze klassieke Hilbert-transformatie hoeven slechts 4 vermenigvuldigingen te worden uitgevoerd. De resulterende getransformeerde uitvoer is in fase met het 8e sample in de array, namelijk I[7], Q[7] en de berekende Qh.

The Q delay line array has a length of 15, to enable a 15-tap classic Hilbert transform. The even samples have a zero coefficient so, as in the above figure, only 8 of the samples are used in the calculation. Due to symmetry of this classic Hilbert transform only 4 multiplications have to be performed. The resulting transformed output is in phase with the 8th sample in the array, being I[7], Q[7] and the calculated Qh.

De coëfficiënten voor de aftakkingen kunnen worden afgeleid uit de Hilbert-transformatie-regels in combinatie met de keuze van een juiste vensterfunctie. De vensterfunctie onderdrukt de rimpel die anders te zien is in de frequentierespons. Zie bijvoorbeeld Iowa Hills tools om een set coëfficiënten te verkrijgen.

The coefficients for the taps can be derived from the Hilbert transform rules combined with the choice of a proper windowing function. The window function suppresses the ripple otherwise seen in the frequency response. See for example Iowa Hills tools to obtain a set of coefficients.

De coëfficiënten worden gegeven door:

$$h(n) = w(n) \cdot \frac{2}{\pi \cdot n}$$

waarbij n oneven is [-7, 7]

The coefficients are given by:

$$h(n) = w(n) \cdot \frac{2}{\pi \cdot n}$$

where n is odd [-7, 7]

In deze functie is w(n) een vensterfunctie, bijvoorbeeld een Hamming (verhoogde cosinus) venster:

In this function w(n) is a windowing function, for example a Hamming (raised cosine) window:

$$w(n) = 0.54 + 0.46 \cdot cos\left(\frac{\pi \cdot n}{7}\right)$$

Ook hier is n weer oneven [-7, 7]

Merk op dat de bandbreedte van de klassieke Hilbert-transformatie de helft van de bemonsteringsfrequentie is. De respons aan de randen neemt af, dus om een respons te krijgen die ver genoeg naar de randen reikt, moet de bemonsteringsfrequentie worden verlaagd of moet het aantal segmenten worden verhoogd. Voor 15 segmenten zou de snelheid ½ van 64usec zijn, ongeveer. 7800Hz.

$$w(n) = 0.54 + 0.46 \cdot cos\left(\frac{\pi \cdot n}{7}\right)$$

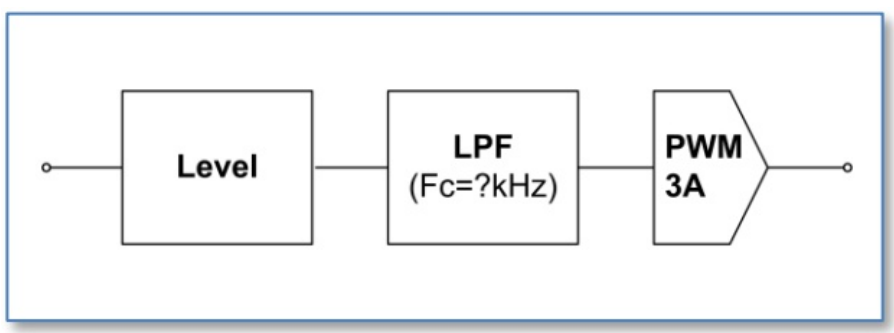
again, n is odd [-7, 7]

Note that the bandwidth of the classic Hilbert transform is half the sampling frequency. The response at the edges drops off, so to get a response that extends far enough towards the edges, either the sampling rate must be lowered or the number of taps must be increased. For 15 taps the rate would be ½ the 64usec, appr. 7800Hz.

-7	-5	-3	-1	1	3	5	7
-0.00728	-0.03224	-0.13631	-0.60762	0.60762	0.13631	0.03224	0.00728

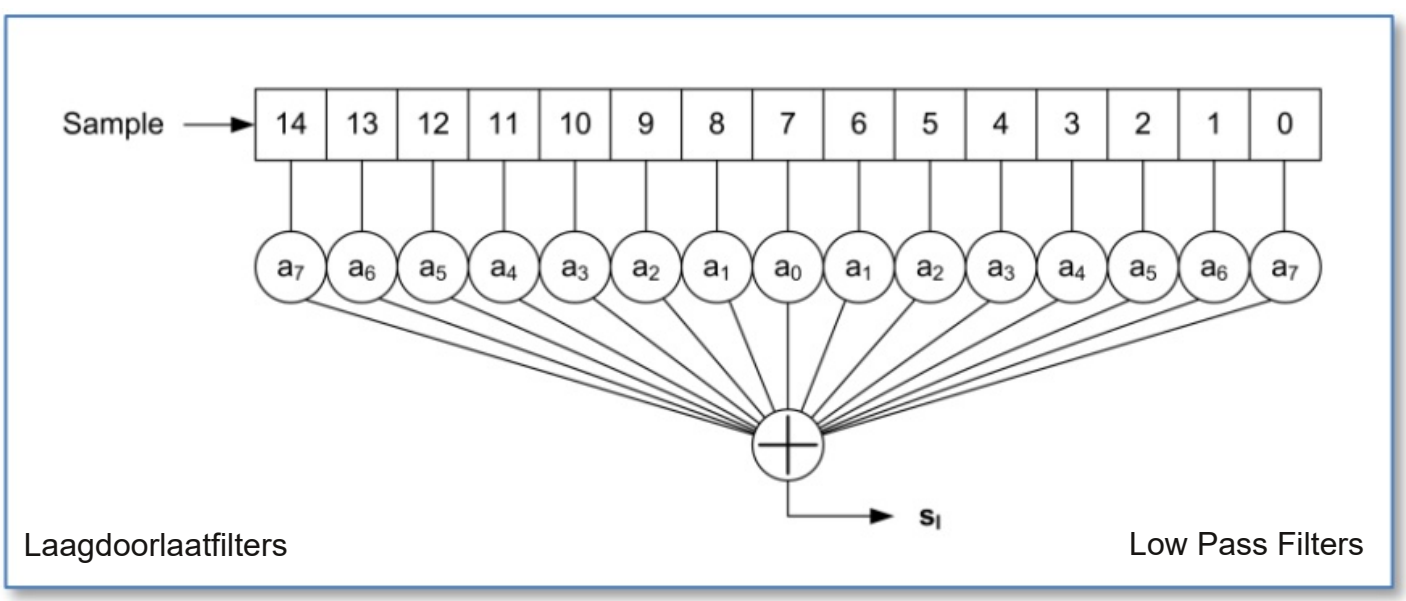
Audio opwekking

Audio generation



De gedemoduleerde audiosamples gaan door een niveaudetector, die een AGC-feedback-isignaal genereert om de uitvoer te schalen naar binnen het PWM/DAC-bereik. De schaling is logaritmisch, in factoren van 2 (stappen van 6 dB), waardoor eenvoudige bitverschuivingen kunnen worden gebruikt als versterking/ verzwakking in het bemonsteringsblok.

The demodulated audio samples pass through a level detector, which generates an AGC feedback signal in order to scale the output to within the PWM/DAC range. The scaling is logarithmic, in factors of 2 (6dB steps), enabling simple bit shifts to be used as amplification/ attenuation in the sampling block.



Laagdoorlaatfilters

Low Pass Filters

Er zijn verschillende laagdoorlaatfilters met 15 segmenten gemaakt, met een afsnijfrequentie van Fc=3kHz. De stopband hangt af van de werkelijke samplefrequentie waarvoor het filter is ontworpen. Dat begint meestal rond 5 kHz, met een niveau van -40dB of beter. De code bevat filters voor 62,5 kHz, 31,25 kHz en 15,625 kHz samplefrequenties, hoewel alleen de eerste en de laatste daadwerkelijk worden gebruikt.

Several 15-tap low pass filters have been created, with a corner frequency of Fc=3kHz. The stop-band depends on the actual sample rate the filter is designed for. It usually starts around 5kHz, with a level of -40dB or better. The code contains filters for 62.5 kHz, 31.25 kHz and 15.625 kHz sample rates, although only first and last are actually used.

Deze laagdoorlaatfilters zijn eenvoudige symmetrische FIR-filters, die de impulsrespons van het gewenste laagdoorlaatgedrag vertegenwoordigen. Ze bestaan uit 15 signed integer-arrays. Daarom moeten er per sample 15 vermenigvuldigingen en optellingen worden gedaan, maar de RP2040 heeft een 32-bits MPY-instructie met een enkele cyclus, dus dat gaat snel genoeg.

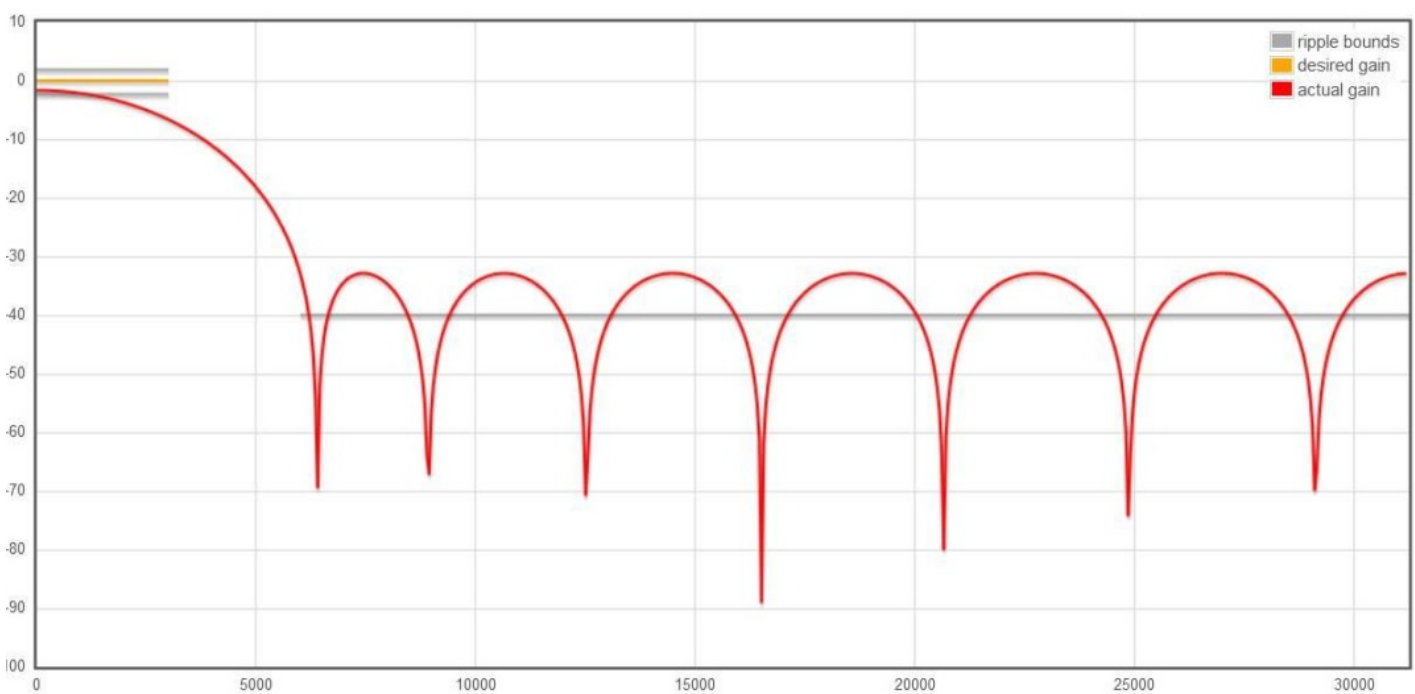
These low pass filters are simple symmetric FIR filters, that represent the impulse response of the desired low pass behavior. They consist of 15 signed integer arrays. Hence, per sample 15 multiplications and additions need to be done, but the RP2040 has a single cycle 32bit MPY instruction, so that be fast enough.

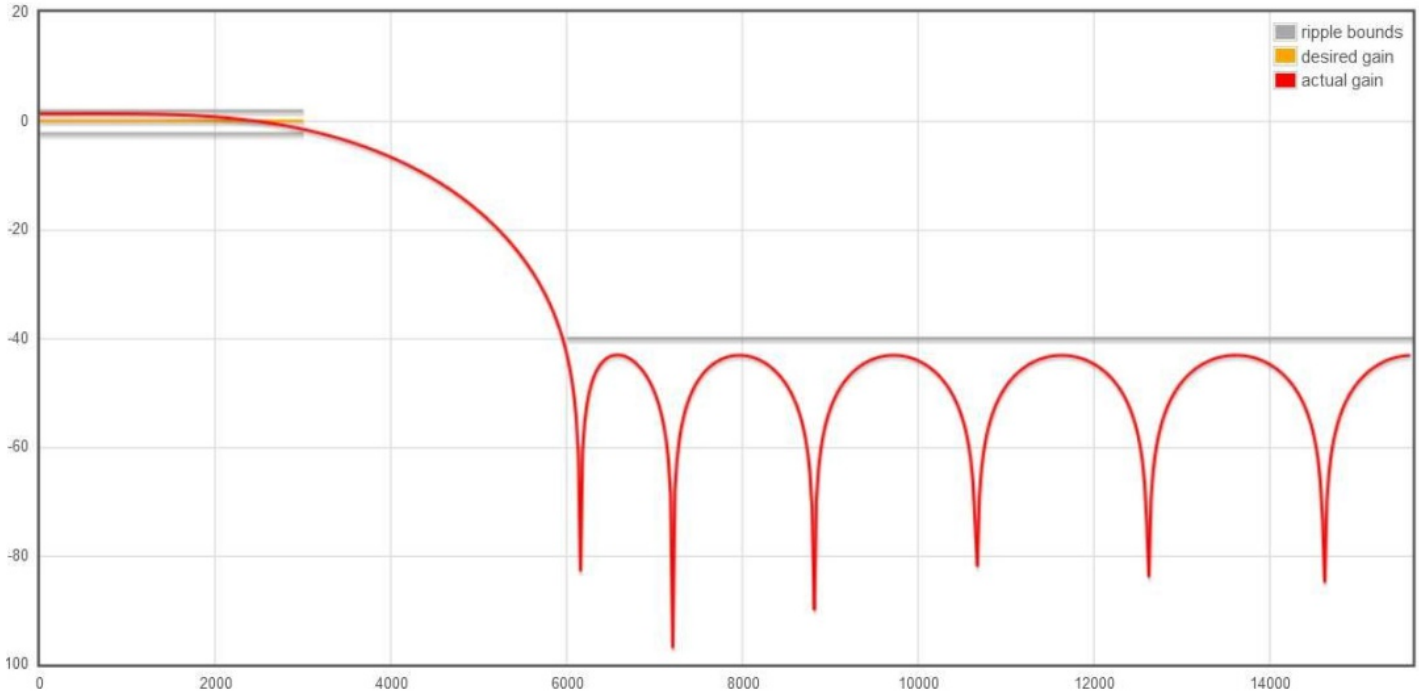
Om de juiste coëfficiënten te vinden, zie bijvoorbeeld Iowa Hills DSP tools of de T-Filter online calculator:
http://www.iowahills.com/
http://t-filter.engineerjs.com/

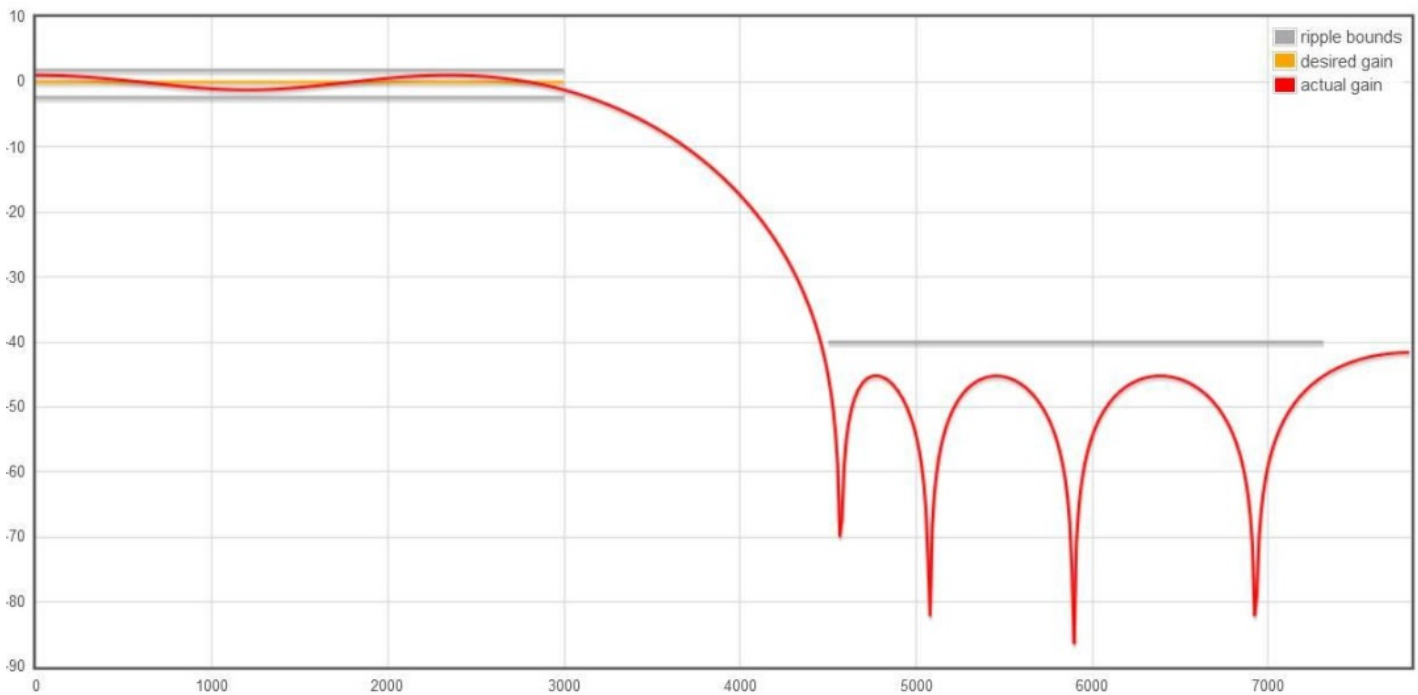
To find the proper coefficients, see for example Iowa Hills DSP tools or the T-Filter on-line calculator:
http://www.iowahills.com/
http://t-filter.engineerjs.com/

T-filter parameters 62500, 31250 en 15625 samplefrequenties, doorlaatband 3kHz rimpel <5dB, stopband van 6kHz bij -40dB:

T-filter parameters 62500, 31250 and 15625 sample rates, passband 3kHz ripple <5dB, stopband from 6kHz at -40dB:



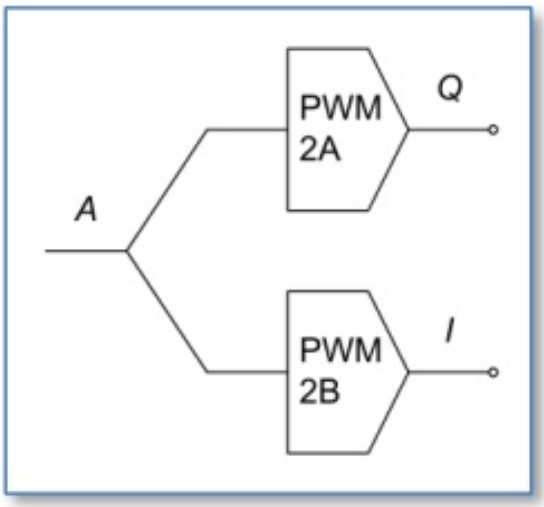




Merk op dat voor het laagdoorlaatfilter van 3 kHz een FIR-algoritme met 15 segmenten het beste werkt bij lagere samplefrequenties. Voor hoge samplefrequenties is het beter om meer segmenten te gebruiken. Momenteel wordt in de uSDR-Pico alleen het filter met lage bemonsteringsfrequentie gebruikt.

Note that for the 3kHz low pass filter, a 15 tap FIR algorithm works best at lower sample rates. For high sample rates it would be better to use more taps. Currently only the low sample rate filter is used in uSDR-Pico.

3.1.2.2 TX stream



De TX-stream is het omgekeerde van de RX-stream: hier zijn dezelfde componenten te vinden als in de RX-stream. Net als bij RX wordt SSB gegenereerd door de I-samples om te zetten in Q-samples via een Hilbert-transformatie met 15 segmenten. De Q-uitgang wordt vermenigvuldigd met -1 voor USB en +1 voor LSB, de I-samples moeten met 7 worden vertraagd.

Het dynamische bereik van de samplestreams komt overeen met dat van het DAC-bereik, voordat het wordt uitgevoerd.

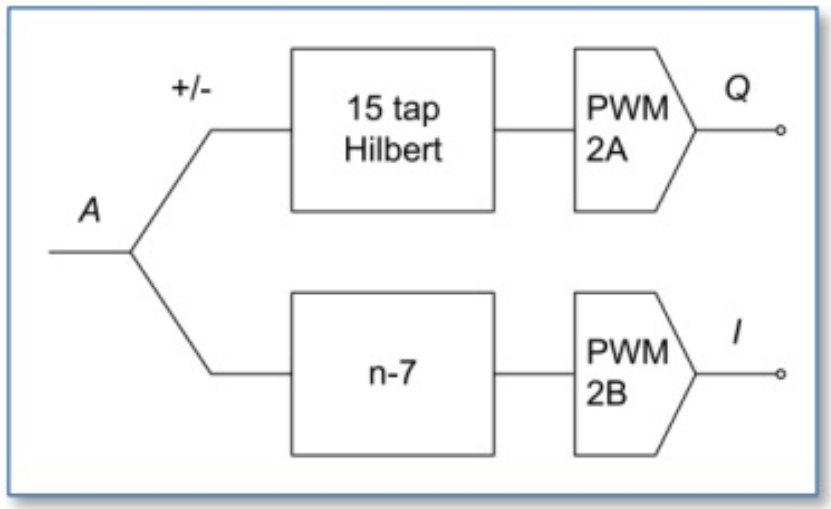
3.1.3 <u>Frequentiedomein signaalverwerking (dsp_fft.c)</u>

Terwijl Time Domain-signaalverwerking wordt uitgevoerd voor elke binnenkomende sample, vereist de FFT een buffer vol met samples en daarom wordt de signaalverwerking aangeroepen telkens wanneer een buffer wordt gevuld. Daarom krijgt de signaalprocessor slechts om de 512 samples (512x64usec ≈ 32msec) een seintje en wordt deze op de achtergrond uitgevoerd terwijl deze wordt onderbroken door de timer met ruwe samplefrequentie.

3.1.3.1 Bufferafhandeling

De bufferstructuur is opgebouwd uit buffers van ½ FFT-grootte, verdubbeld voor de complexe kant van de verwerking. Bufferoverlapping wordt gebruikt om een soepele hereniging van de

3.1.2.2 TX stream



The TX stream is the inverse of the RX stream: the same components can be found here as in the RX stream. As in the RX the SSB is generated by converting the I-samples into Q-samples through a 15 tap Hilbert transform. The Q output is multiplied by -1 for USB and +1 for LSB, the I-samples need to be delayed by 7.

The dynamic range of the sample streams is matched with that of the DAC range, before output.

3.1.3 <u>Frequency domain signal processing (dsp_fft.c)</u>

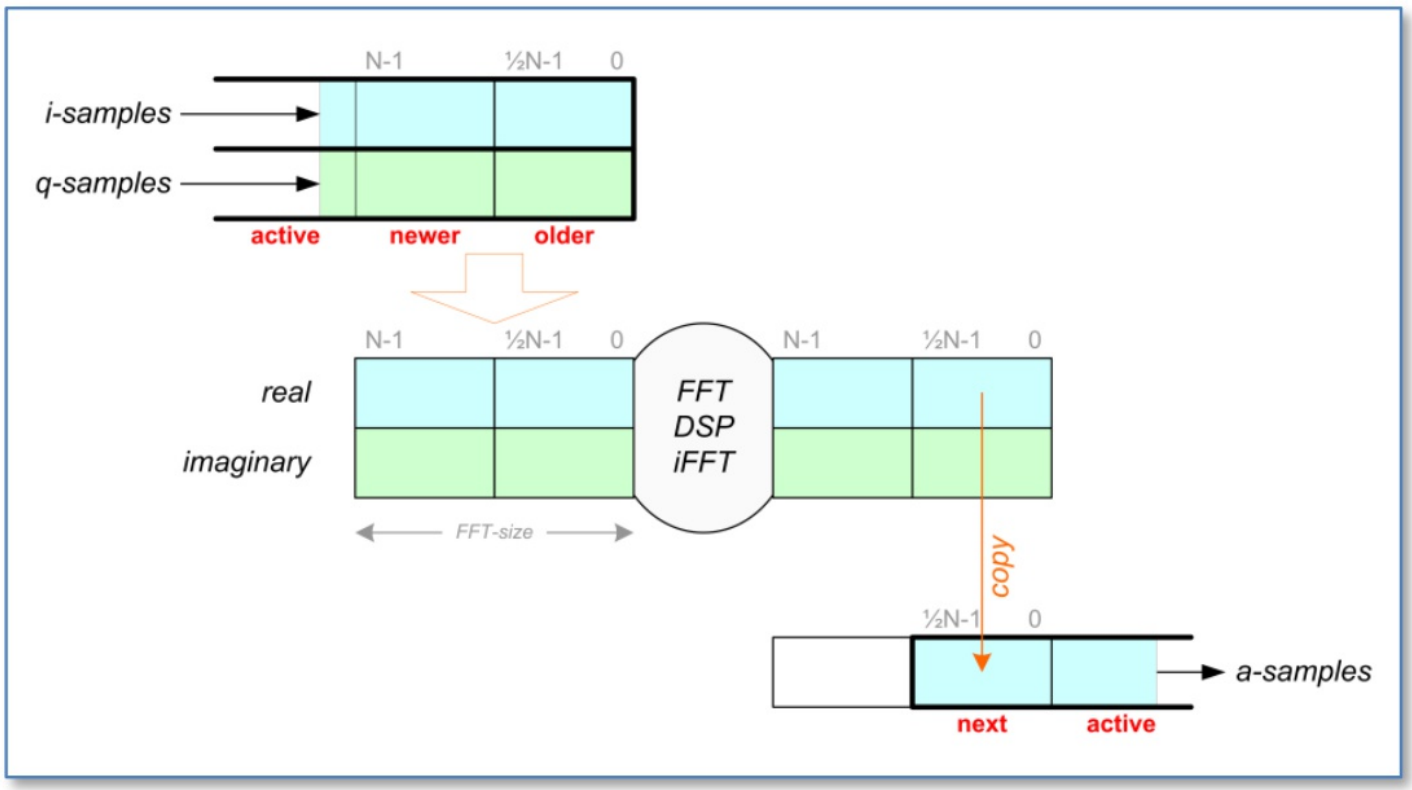
While Time Domain signal processing is done for every incoming sample, the FFT requires a buffer full of samples and hence the signal processing is invoked every time a buffer is filled. Therefore, the Signal Processor is signalled only every 512 samples (512x64usec ≈ 32msec), and runs in the background while interrupted by the timer at raw sampling rate.

3.1.3.1 Buffer handling

The buffer structure is built up from ½ FFT-size buffers, doubled for the complex side of processing. Buffer overlapping is used to ensure a smooth glueing of the chopped-up sample

opgeknipte samplestromen te garanderen. De afhandeling vindt plaats binnen de timer-callback-routine.

streams. The handling is done inside the timer callback routine.



Bovenstaande figuur geeft de RX situatie weer, de toegewezen buffers worden in feite hergebruikt voor de TX situatie, maar werken in tegengestelde richting. De actieve interface-buffer is er één van een wachtrij met 3 buffers, de andere bevat de opgeslagen monsters van het vorige interval. De actieve buffers verzamelen de I- en Q-samples die zijn verzameld door de timer-callback-routine. Wanneer de actieve buffer vol is, worden de invoer- en uitvoerbuffers gereorganiseerd en krijgt de DSP-loop een seintje om te starten.

The picture above represents the RX case, the allocated buffers are in fact re-used for the TX case but work in opposite direction. The active interface buffer is one of a 3-buffer queue, the other being the saved samples of previous interval. The active buffers collect the I and Q samples captured by the timer callback routine. Whenever the active buffer is full, the input and output buffers are reorganized and the DSP loop is signaled to start.

Het reële deel van het vorige resultaat van de signaalverwerkingscyclus wordt gekopieerd naar de uitgangsbuffers. Vervolgens worden de opgeslagen invoerbuffers gekopieerd naar de onderste helft van de FFT-buffers, terwijl de bovenste helften aangevuld zijn met nullen. Vervolgens wordt de signaalverwerkingscyclus gestart, wat een nieuw resultaat oplevert.

The real part of the previous signal processing cycle result is copied to the output buffers. Then the saved input buffers are copied into the lower half of the FFT buffers, while the upper halves are zero padded. Then the signal processing cycle is begun, yielding a new result.

3.1.3.2 Signaalverwerking

3.1.3.2 Signal processing

De signaalverwerkingsengine volgt de volgorde

The signal processing engine follows the

van transformatie naar het frequentiedomein en past de bandfiltering en -verschuiving toe, evenals transformatie terug naar het tijdsdomein. De samples in het audiodomein zijn reëel, dus een conversie van (en naar) de complexe representatie is vereist.

RX-situatie:

<<Shift & Filter>>

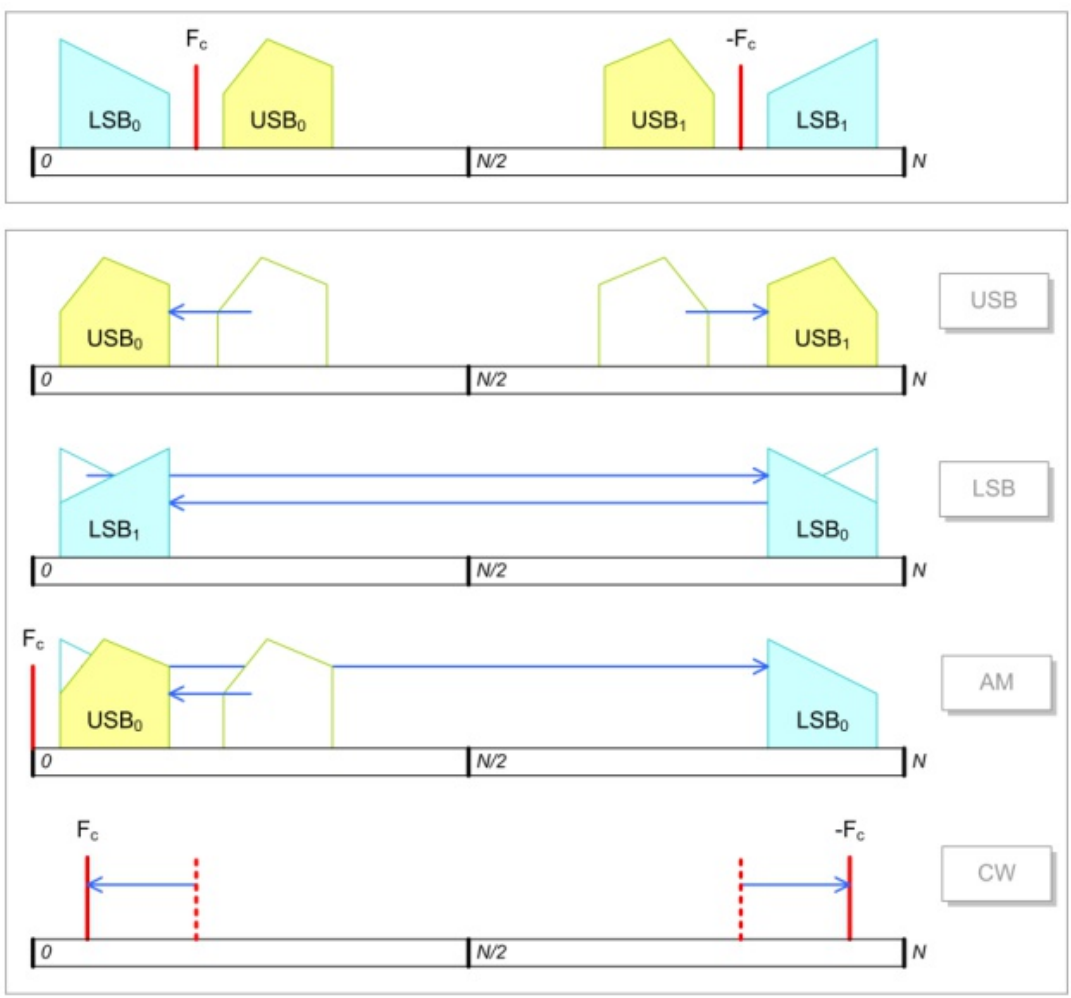
Om de filtering mogelijk te maken, moet de draaggolffrequentie niet omlaag worden geconverteerd naar 0 Hz, maar naar ergens in het midden van de frequentieband die resulteert uit de FFT. Met een bemonsteringsfrequentie van 15,625 kHz zou de offset-frequentie Fc ergens rond de 3,9 kHz moeten liggen. Afhankelijk van de gewenste modulatiemethode worden verschillende bereiken met betrekking tot deze offset uitgefilterd en naar de juiste plaats in de spectrumbuffer verschoven. Bijvoorbeeld (alleen reëel spectrum getoond):

sequence of transformation to the frequency domain, applying the band filtering and shifting as well as transformation back to the time domain. The samples in the audio domain are real, so a conversion from (and to) the complex representation is required.

RX case:

<<Shift & Filter>>

To enable the filtering, the carrier frequency must not be downconverted to 0 Hz, but rather to somewhere in the center of the frequency band resulting from the FFT. With a sampling rate of 15.625kHz, the offset frequency Fc should be somewhere around 3.9kHz. Depending on the desired modulation mode, different ranges with respect to this offset are filtered out, and shifted to the proper place in the spectrum buffer. For example (only real spectum shown):



Voor AM bevatten de bovenste en onderste zijbanden dezelfde informatie, d.w.z. het spectrum rond Fc is symmetrisch. Dit houdt in dat de bijbehorende zijbanden kunnen worden gespiegeld en opgeteld voor een versterking van 3 dB.

Voor CW kan het filter smal zijn rond Fc en moet de effectieve verschuiving worden verminderd met de gewenste toon (bijvoorbeeld 900 Hz). Na de iFFT van dit gefilterde spectrum wordt het reële deel van de complexe tijdsamples gekopieerd naar de buffer voor audiosamples.

TX-situatie:

De omgekeerde acties worden uitgevoerd bij zenden. De audiosamples worden gekopieerd in het reële deel van de FFT-buffer, terwijl het imaginaire deel op 0 gezet wordt. Na het uitvoeren van de FFT wordt het spectrum omhoog geschoven met de offset, wordt het gewenste spectrum eruit gefilterd en de iFFT uitgevoerd. Zowel reële als imaginaire delen worden gekopieerd naar de I- en Q-buffers.

Opmerkingen:

Bij het verschuiven van het spectrum wordt in feite een rotatie gedaan; segmenten die uit de FFT-buffer schuiven, komen aan de andere kant weer binnen.

Bij het toevoegen van een draaggolf om een AM-basisbandsignaal te verkrijgen, moet deze draaggolf twee keer (?) de amplitude van een zijbandsignaal bevatten

3.1.4 Fast Fourier-transformatie (fix_fft.c)

De crux van de signaalverwerking in het frequentiedomein is de toepassing van de Fast Fourier Transform (FFT). Deze procedure zet de tijdsamples om in frequentiebakken en vice versa.

3.1.4.1 Fysieke betekenis van de FFT

Een HF-signaal wordt omlaag gemengd met een

For AM the upper and lower sidebands contain the same information, i.e. the spectrum about the Fc is symmetric. This implies that the corresponding sidebands could be mirrored and added for a 3dB gain.

For CW the filter can be narrow around Fc and the effective shift should be reduced with the desired tone (e.g. 900Hz). After the iFFT of this filtered spectrum, the real part of the complex time samples are copied to the audio samples buffer.

TX case:

The reverse actions are performed for transmission. The audio samples are copied in the real part of the FFT buffer, while the imaginary part is set to 0. Then after performing the FFT shift the spectrum up with the offset, filter out the desired spectrum and do the iFFT. Both real and imaginary parts are copied to the I and Q buffers.

Notes:

When shifting the spectrum, in fact a rotation is done; bins that shift beyond the FFT-buffer edge will re-enter on the other side.

When adding a carrier to obtain an AM baseband signal, this carrier should contain twice (?) the amplitude of any sideband signal

3.1.4 Fast Fourier Transform (fix_fft.c)

The crux of the frequency domain signal processing is the application of the Fast Fourier Transform (FFT). This procedure transforms the time samples into frequency bins and vice versa.

3.1.4.1 Physical meaning of the FFT

An RF signal is mixed down with a direct

directe conversie kwadratuurmixer, wat resulteert in een in-fase (I) en een kwadratuur (Q) basisband, gecentreerd rond DC. Dus wat betekent dit, bijvoorbeeld om negatieve frequenties te hebben? Als je bedenkt dat het originele HF-signaal twee zijbanden heeft door amplitude-modulatie, dan zijn de zijbanden net iets hoger of lager dan de draaggolffrequentie. Wanneer je dit signaal omlaag mengt met de draaggolf-frequentie, wordt de onderste zijband door het wiskundig resultaat weergegeven als een negatieve frequentie.

conversion quadrature mixer, resulting in an in-phase (I) and a quadrature (Q) baseband, centered on DC. So what does this mean, for example to have negative frequencies? If you consider the original RF signal having two sidebands from amplitude modulation, the sidebands are just a bit higher or lower than the carrier frequency. When you mix this signal down with the carrier frequency, the lower sideband as a mathematical consequence is represented by a negative frequency.

Stel dat het omlaag gemengde signaal wordt weergegeven door de tijdafhankelijke vector (It,Qt). De rotatiesnelheid van de vector vertegenwoordigt de frequentie (DC roteert helemaal niet), de rotatierichting geeft aan of de frequentie positief of negatief is. De werkelijke beweging van de vector is grillig en bevat een superpositie van frequenties. Met de fouriertransformatie willen we eigenlijk deze set van rotatiesnelheden (frequenties) analyseren en bepalen in hoeverre deze aanwezig zijn in de (It,Qt) sample stroom.

Suppose that the mixed down signal is represented by the time dependent vector (It,Qt). The rotation speed of the vector represents the frequency (DC doesn't rotate at all), the rotation direction represents whether the frequency is positive or negative. The actual movement of the vector is erratical, and contains a superposition of frequencies. With the fourier transform we actually want to analyze this set of rotation speeds (frequencies) and determine to what extent these are present in the (It,Qt) sample stream.

Hiertoe worden de I- en Q-stromen geconverteerd naar discrete (Ik, Qk) monsterparen, die de complexe tijdsdomeininvoer voor de FFT zijn. Op regelmatige tijdstippen worden de laatste N-samples (d.w.z. de FFT-grootte) omgezet in een frequentiespectrum. Deze transformatieperiode moet korter zijn dan wat wordt weergegeven door N monsters, dus er is enige overlap.

To this purpose, the I and Q streams are converted in discrete (Ik,Qk) sample pairs, which are the time-domain complex input to the FFT. At regular moments in time the latest N samples (i.e. the FFT size) are transformed into a frequency spectrum. This transformation period has to be shorter than what is represented by N samples, so there is some overlap.

2N real-time samples zouden resulteren in N complexe frequenties (zie Nyquist), waarbij de ontbrekende negatieve complexe frequenties slechts een spiegel zijn van de N positieve frequentiesegmenten. Daarentegen resulteren 2N complexe tijdsamples in N positieve + N negatieve complexe frequenties.

2N real time samples would result in N complex frequencies (cf. Nyquist), where the missing negative complex frequencies are just a mirror of the N positive frequency bins. In contrast, 2N complex time samples result in N positive + N negative complex frequencies.

Het segment met index 0 vertegenwoordigt de DC-component en het segment met index N-1 vertegenwoordigt de Nyquist-frequentie. De segmenten N en verder vertegenwoordigen de negatieve frequenties en segment 2N zou weer

The bin with index 0 represents the DC component, and the bin with index N-1 represents the Nyquist frequency. The bins N and beyond represent the negative frequencies and bin 2N would be DC again. For

DC zijn. Voor representatie zal het draaien van de set frequentiesegmenten met N de DC-component op segment N plaatsen met de bovenste/onderste zijbanden aan weerszijden.

representation, rotating the set of frequency bins with N will place the DC component at bin N and the upper/lower sidebands on either side.

Demodulatie van SSB zou neerkomen op het wegfilteren van alles behalve de 3 kHz of zo aan de hoge kant van het DC-segment voordat het terug naar het tijdsdomein wordt getransformeerd. Aangezien de bovenste en onderste zijbanden verschillend zijn, is het duidelijk dat complexe tijdsamples nodig zijn om de juiste transformatie te verkrijgen.

Demodulation of SSB would boil down to filtering away everything but the 3kHz or so on the high side of the DC bin before transforming back to the time domain. Since upper and lower sidebands are different, it is clear that complex time samples are required to obtain the right transformation.

Om ruis rond DC te verwijderen, zou de QSD-mengfrequentie lager kunnen worden gekozen dan de werkelijke HF-draaggolf, waardoor het basisbandsignaal ergens in het midden van de positieve frequentiesegmenten terechtkomt, d.w.z. rond N/2. Na het filteren kan de basisband met N/2 naar beneden worden verschoven voordat de inverse FFT wordt toegepast.

To get rid of noise around DC, the QSD mixing frequency could be chosen lower than the actual RF carrier, causing the baseband signal to land somewhere in the middle of the positive frequency bins, i.e. around N/2. Then after filtering, the baseband can be shifted down by N/2 before applying the inverse FFT.

3.1.4.2 FFT-implementatie

3.1.4.2 FFT implementation

Het FFT-algoritme wordt nader toegelicht in een bijlage.

The FFT algorithm is explained in more detail in an appendix.

De eerste fase van het FFT-algoritme is het opnieuw ordenen van de tijddomeinsamples; om precies te zijn, de samples met indexen die bit-reverse van elkaar zijn, moeten worden verwisseld. De arraygrootte is 1024 samples, dus de index is 10 bits. De swap is dan bijvoorbeeld tussen monsters [1111000001b] en [1000001111b]. Deze herordening wordt gedaan voordat de FFT wordt berekend, en daarom Decimation in Time (DIT) genoemd, in tegenstelling tot de post-FFT DIF. De herschikking is nodig om een efficiënte keten van butterfly executies mogelijk te maken.

The first stage of the FFT algorithm is the reordering of the time domain samples; to be precise, the samples with indexes that are bit-reverse of each other need to be swapped. The array size is 1024 samples, so the index is 10 bits. The swap is then for example between samples [1111000001b] and [1000001111b]. This re-ordering is done before the FFT is calculated, and therefore named Decimation in Time (DIT), as opposed to the post-FFT DIF. The re-ordering is needed to enable an efficient chain of butterfly executions.

Het punt van het algoritme is hoe het slechts één keer door de array gaat, d.w.z. vermijd het opnieuw verwisselen van samples. Een algemene benadering zou zijn:

The point of the algorithm is how it goes through the array only once, i.e. avoid swapping samples back again. A general approach would be:

```
for (i=0; i<1024; i++)
{
    j = bit_reverse(i);
    if (i < j)
        swap(data[i], data[j]);
}
```

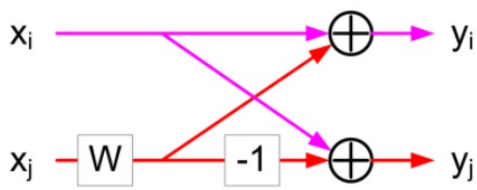
```
for (i=0; i<1024; i++)
{
    j = bit_reverse(i);
    if (i < j)
        swap(data[i], data[j]);
}
```

maar de bit_reverse-routine kan behoorlijk wat tijd in beslag nemen. Een sneller alternatief (gebruik makend van het relatief overvloedige RAM) is om in plaats daarvan een opzoektabel te gebruiken. Daarna is het omwisselen van de samples eenvoudig.

but the bit_reverse routine could take quite some time. A faster alternative (utilizing the relatively abundant RAM) is to use a lookup table instead. After that, the swapping of the samples is straightforward.

De tweede fase bestaat uit een aantal geneste loops, voor het berekenen en toevoegen van de butterflies. Een dergelijke butterfly kan als volgt worden weergegeven:

The second stage consists of a number of nested loops, calculating and adding the butterflies. One such butterfly can be represented as follows:



Eén (complexe) ingang wordt vermenigvuldigd met een "Wiggle-factor" en afgetrokken van of opgeteld bij de andere ingang. Dit wordt herhaald over de volledige array in 2log(N) opeenvolgende fasen, waarbij de combinaties en W-factoren per fase veranderen. Het resultaat van elke fase wordt op dezelfde locatie opgeslagen, vandaar de notatie "in-place".

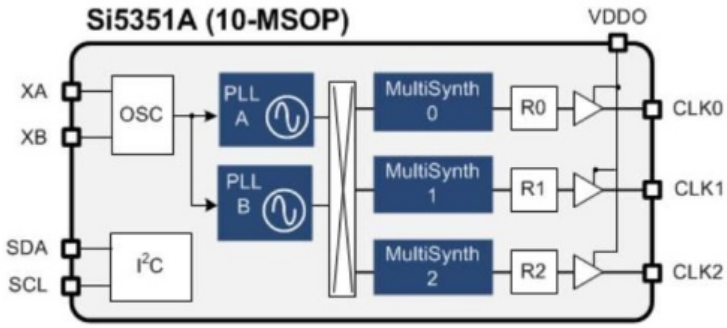
One (complex) input is multiplied with a “Wiggle factor” and either subtracted from or added to the other input. This is repeated over the complete array in 2log(N) subsequent stages, where the combinations and W factors change per stage. The result of each stage is stored in the same location, hence the notation “in-place”.

De FFT-functie kan op twee manieren worden uitgevoerd, vooruit en achteruit, die bijna identiek zijn, op een factor na. In principe zou het na elkaar toepassen van de twee manieren resulteren in de originele samples.

The FFT function has two ways to execute, in forward and in reverse, which are almost identical, except for a factor. In principle applying one after the other would result in the original samples.

3.2 Kwadratuur VFO (si5351.c)

3.2 Quadrature VFO (si5351.c)



De Si5351A is een drievoudige klokgenerator, die kan worden bestuurd via de I²C-interface. Er zijn drie klokuitgangstrappen, die kunnen worden aangestuurd door twee PLL's. Deze PLL's vermenigvuldigen de kristaloscillatorfrequentie (meestal 25 MHz) met een bepaalde hoeveelheid, waarvan de klok uitgangen wordt afgeleid door een andere vermenigvuldiging (deling). Ook kan een faseverschuiving aan een klokuitgang worden gegeven, en wanneer twee klokken afgeleid zijn van dezelfde PLL, is de faserelatie deterministisch. Deze karakteristiek wordt gebruikt om een kwadratuur VFO te maken, met twee uitgangen met dezelfde frequentie maar met gecontroleerd faseverschil (0, 90, 180 of 270 graden). Voor de Q-mixeringang is een sin() nodig die een fase van -90˚ heeft ten opzichte van de I-mixeringang, wat een cos()-signaal is. Je kunt ook zeggen dat de sin() een kwartgolf vertraagde cos() is.

The Si5351A is a triple clock generator, that can be controlled through I²C interface. There are three clock output stages, that can be driven by two PLLs. These PLLs multiply the crystal oscillator frequency (usually 25MHz) by some amount, from which the clock outputs are derived by another multiplication (division). Also, a phase offset can be given to a clock output, and when two clocks rely on the same PLL, the phase relation is deterministic. This characteristic is used to make a quadrature VFO, with two outputs with the same frequency but with controlled phase difference (0, 90, 180 or 270 degree). For the Q mixer input, a sin() is needed which has a -90˚ phase with regard to the I mixer input, which is a cos() signal. You can also say that the sin() is a quarter wave delayed cos().

De fractionele vermenigvuldiger voor de PLL-trap moet zo zijn dat de resulterende frequentie tussen 600 en 900 MHz ligt (MSN ligt tussen 24 en 32). Deze grenzen zijn niet erg moeilijk en kunnen logischerwijs tussen 15 en 90 liggen, maar laten we ons voorlopig aan het voorgeschreven bereik houden. De Multisynth fractionele deler voor klok i is MSi (8..2048), waarna een extra deling met een integer (heel getal) factor Ri (1..128) volgt.

The fractional multiplier for the PLL stage must be so, that the resulting frequency is between 600 and 900MHz (MSN is between 24 and 32). These boundaries are not very hard, and can logically be between 15 and 90, but for the moment let’s stick to the prescribed range. The Multisynth fractional divider for clock i is MSi (8..2048), after which an additional division with an integer factor Ri (1..128).

De vermenigvuldiger en deler worden geschreven als: a+b/c

The multiplier and divider are written as: a+b/c

De truc is nu om de integer-modus voor MSi te gebruiken, wat betekent dat dit een (even) integer-deling zou moeten zijn. Alleen dan kan de fase offset worden gebruikt om een exact faseverschil te produceren.

The trick is now to use integer mode for MSi, meaning that this should be an (even) integer division. Only then the phase offset can be used to produce an exact phase difference.

Dus vanaf mid-range PLL-uitgang (750 MHz), kun je MSi en Ri instellen om in de marge van de gewenste uitgangsfrequentie te komen. Dan kan het afstemmen worden gerealiseerd door de PLL-multiplicator MSN te wijzigen:

- $F_{out} = F_{vco} / (MSi*Ri)$
- $F_{vco} = F_{xo} * MSN$

So starting from mid-range PLL output (750MHz), you can set MSi and Ri to get into the ballpark desired output frequency. Then tuning can be done by changing the PLL multiplicator MSN:

- $F_{out} = F_{vco} / (MSi*Ri)$
- $F_{vco} = F_{xo} * MSN$

Enkele uitersten van het bereik (varieer MSi om iets daartussenin te krijgen):

Some range extremes (vary MSi to get anything between):

Ri	MSi	Range [MHz]
1	4	150.000 – 225.000
1	126	4.762 – 7.143
32	4	4.688 – 7.031
32	126	0.149 – 0.223
128	4	1.172 – 1.758
128	126	0.037 – 0.056

In de praktijk gebruiken we:

- Ri=128 voor F_{out} <1 MHz
- Ri=32 voor F_{out} 1-6 MHz
- Ri=1 voor F_{out} >6 MHz

In practise we use:

- Ri=128 for F_{out} <1 MHz
- Ri=32 for F_{out} 1-6 MHz
- Ri=1 for F_{out} >6 MHz

Er zijn twee VFO's gedefinieerd, VFO 0 (uitgang op clk0 en clk1) en VFO 1 (uitgang op clk2). Er zijn een aantal macro's gedefinieerd om de vfo te besturen:

Two VFOs have been defined, VFO 0 (output on clk0 and clk1) and VFO 1 (output on clk2). A number of macro's have been defined to control the vfo:

- `SI_GETFREQ(i)` Returns frequency of VFO i
- `SI_INCFREQ(i, d)` Increment frequency of VFO i with d Hz
- `SI_DECFREQ(i, d)` Decrement frequency of VFO i with d Hz
- `SI_SETFREQ(i, f)` Set frequency of VFO i to f Hz
- `SI_SETPHASE(i, p)` Set phase delay of VFO i to (p = {0, 1, 2, 3} x 90deg)

Opmerking: SI_SETPHASE werkt uiteraard alleen voor VFO 0, waar de vertraging is geïntroduceerd in clk1.

Note: SI_SETPHASE obviously only works for VFO 0, where the delay is introduced in clk1.

De functie si_evaluate() wordt aangeroepen om te evalueren of VFO-instellingen daadwerkelijk zijn gewijzigd en vervolgens de nieuwe instellingen naar de si5351-registers te schrijven. Dat is efficiënter dan schrijven voor elke Hz bij het draaien aan de afstemknop.

The function si_evaluate() is called to evaluate whether VFO settings have actually changed and then write the new settings to the si5351 registers. That is more efficient than writing for every Hz when turning the tuning knob.

3.3 Display (lcd.c)

Het display is een 16x2 LCD die wordt aangestuurd met de bekende HD44780-chip, maar de hier gebruikte versie wordt aangestuurd via een I²C-bus. Dit maakt het mogelijk om in plaats daarvan ook bijvoorbeeld een OLED grafisch display te gebruiken.

The display is a 16x2 LCD controlled with the familiar HD44780 chip, but the version used here is controlled over an I²C bus. This allows to also use for example an OLED graphical display instead.

De software driver bevat een buffer van 16x2 bytes, die indien nodig in twee schrijfacties naar het LCD-scherm kan worden gekopieerd. Natuurlijk kunnen ook karakters achter elkaar worden geschreven. Ook wordt de huidige cursorpositie behouden met de buffer, dit zou normaal gesproken moeten overeenkomen met de cursorpositie op het scherm.

The software driver contains a 16x2 byte buffer, which can be copied to the LCD in two write actions, when necessary. Of course, also characters can be written one after another. Also, the current cursor position is maintained with the buffer, this should normally match the cursor position on screen.

Naast de initialisatiefunctie zijn er verschillende andere functies beschikbaar om het display te besturen:

Apart from the initialization function, there are several other functions available to control the output:

- `lcd_ctrl()` Controls display state.
- `lcd_put()` Output one byte to current cursor position.
- `lcd_write()` Output string to current cursor position.

De uitvoerfuncties verplaatsen de cursorpositie ook horizontaal, totdat de laatste kolom is bereikt.

The output functions also move the cursor location horizontally, until the last column is reached.

De besturingsfunctie ondersteunt de volgende acties:

The control function supports the following actions:

- `LCD_CLEAR` Clear display, cursor to left top position
- `LCD_HOME` Cursor to left top position
- `LCD_GOTO` Move cursor to x, y position
- `LCD_CURSOR` Set cursor visible or not
- `LCD_BLINK` Set cursor blinking or not

3.4 Gebruikersinterface (hmi.c)

De gebruikersinterface is eventgestuurd en georganiseerd rond een IRQ-callback routine. Deze handler reageert op events op de GPIO-pinnen die worden gebruikt voor de encoder, voor de knoppen en voor de PTT. De interrupts worden veroorzaakt door stijgende en dalende flanken die op de GPIO worden gedetecteerd.

3.4 User interface (hmi.c)

The user interface is event driven, and organized around an IRQ callback routine. This handler catches the events on the GPIO pins used for the encoder, for the buttons and for the PTT. The interrupts are caused by rising and falling edges detected on the GPIO. From this the encoder increment and decrement events as

Hieruit worden de events voor het verdraaien van de encoder en de events voor het indrukken van de andere knoppen afgeleid.

Events:

- Encoder toename
- Encoder afname
- Enter toets
- Escape toets
- Links toets
- Rechts toets
- PTT ingedrukt
- PTT losgelaten

De HMI kan zich in verschillende toestanden bevinden en afhankelijk van de toestand zullen de events verschillende effecten hebben. Het hoogste niveau is de normale operationele status, die alleen het afstemmen van de werkfrequentie mogelijk maakt. Vanaf dit hoofdniveau kan het submenuniveau worden geopend door op de ESC-knop te drukken. Er is slechts één submenuniveau.

In de submenu's kan een waarde worden gewijzigd met de Encoder en geaccepteerd met Return. Met de Links- en Rechtsknoppen blader je door de submenu's en door nogmaals op ESC te drukken, verlaat je het submenuniveau.

Submenu Statussen (uit te breiden):

• Modus (USB, LSB, AM, CW)
• AGC (Snel, Langzaam, Uit)
• Voorversterker (+10dB, 0dB, -10dB, -20dB, -30dB)

Zenden

Het PTT active event activeert het TX-pad. Het PTT event is direct geassocieerd met het hardware PTT-signaal, dat ook direct hardware bestuurt zoals het Band Pass Filter.

well as the key-pressed events for the other buttons are deduced.

Events:

- Encoder increment
- Encoder decrement
- Enter key
- Escape key
- Left key
- Right key
- PTT activated
- PTT released

The HMI can be in several states, and depending on the state the events will have different effects. The top level is the normal operational state, which only allows tuning the operating frequency. From this top-level the sub-menu level can be entered by pressing the ESC button. There is only one sub-menu level.

In the sub menus a value can be changed with the Encoder and accepted with Return. Left and Right buttons cycle through the sub menus and pressing ESC again exits the sub-menu level.

Submenu States (to be expanded):

• Mode (USB, LSB, AM, CW)
• AGC (Fast, Slow, Off)
• Pre amp (+10dB, 0dB, -10dB, -20dB, -30dB)

Transmission

The PTT active event enables the TX path. The PTT event is directly associated with the HW PTT signal, that also directly controls HW like the BPF.

3.5 Command shell (monitor.c)

De command shell biedt een command line interface op stdin/stdout. De Pico ondersteunt twee opties voor stdio, ofwel naar de USB of naar de fysieke UART0, selecteerbaar in CMakeLists.txt. In de uiteindelijke situatie is het de bedoeling om via de UART in een goede seriële interface te voorzien. Dit ondersteunt dan ook het loggen van fouten tijdens de opstartfase van het apparaat.

The command shell provides a command line interface on stdin/stdout. The Pico supports two mappings for stdio, either to the USB or to the physical UART0, selectable in CMakeLists.txt. In the final situation the idea is to provide a proper serial interface through the UART. This then also supports logging errors during the device start-up phase.

Om stdio in te schakelen, moet een call worden gedaan naar stdio_init_all(). Dit gebeurt in feite binnen de initialisatieroutine van de monitor, dus je kunt dit het beste vroeg in het opstartproces aanroepen.

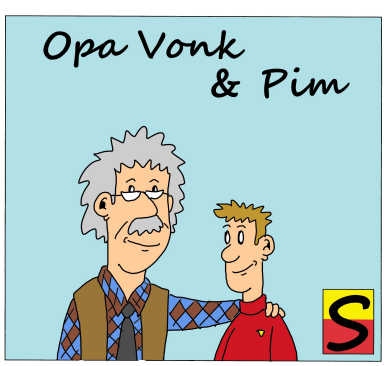
To enable stdio, a call has to be made to stdio_init_all(). This is actually done inside the monitor initialization routine, so best to call this early in the start-up sequence.

De shell-vocabulaire is opgenomen in een array van strings. Telkens wanneer een CR/LF wordt gedetecteerd op stdin, worden de verzamelde tekens behandeld als een opdrachtregel en wordt geprobeerd de strings in de shell-array te matchen. Wanneer een overeenkomst wordt gevonden, wordt de bijbehorende handler aangeroepen, met de rest van de command line. Handlers kunnen waar nodig worden toegevoegd voor foutopsporing of controledoeleinden.

The shell vocabulary is contained in an array of strings. Whenever a CR/LF is entered on stdin, the collected characters are treated as command-line, and a match is attempted with the strings in the shell array. When a match is found, the corresponding handler is invoked, with the remainder of the command-line. Handlers can be added where needed, for debugging or control purposes.

Wordt vervolgd

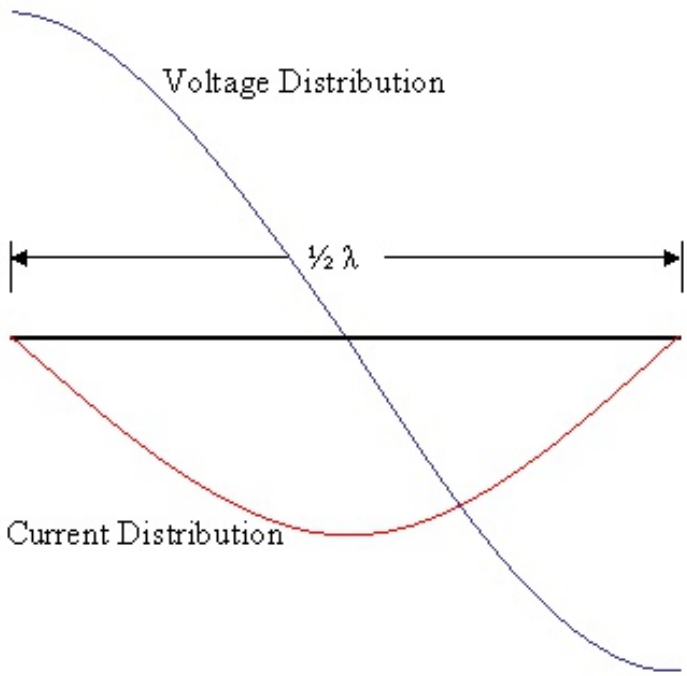
To be continued



Opa Vonk keek naar zijn kleinzoon Pim, die in de weer was met een magnetische loop antenne. Maar bij het volgen van de antenneleiding van de loop naar de ontvanger fronste Opa zijn wenkbrauwen. Pim zag het, en herkende het als de introductie voor een preek van zijn Opa. "Wat doet die tuner daar?" vroeg Opa. Pim volgde Opa's blik en zei: "Die is voor de finetuning", antwoordde Pim. "Ik stem ergens in de band af, maar als ik dan de frequentie verander, moet ik elke keer naar de loop toe lopen om 'm weer af te stemmen omdat de bandbreedte van de loop zo smal is. Met de tuner ertussen stem ik die misaanpassing wel weg met de tuner, dan hoef ik niet meer naar de loop toe". Opa's ogen werden groot van schrik. "Maar dat kan helemaal niet!" riep hij uit. "Dat gaat niet werken!". "Nou, het gaat anders prima", zei Pim. "Ik krijg de SWR keurig 1:1 hoor." "Dat geloof ik wel", zei Opa. "Maar de efficiency van de antenne is weg. Ik heb je al vaker uitgelegd dat een tuner niet de antenne afstemt, maar het antenne*systeem*. Dat is alles wat achter de tuner zit, dus kabel, eventuele baluns, de

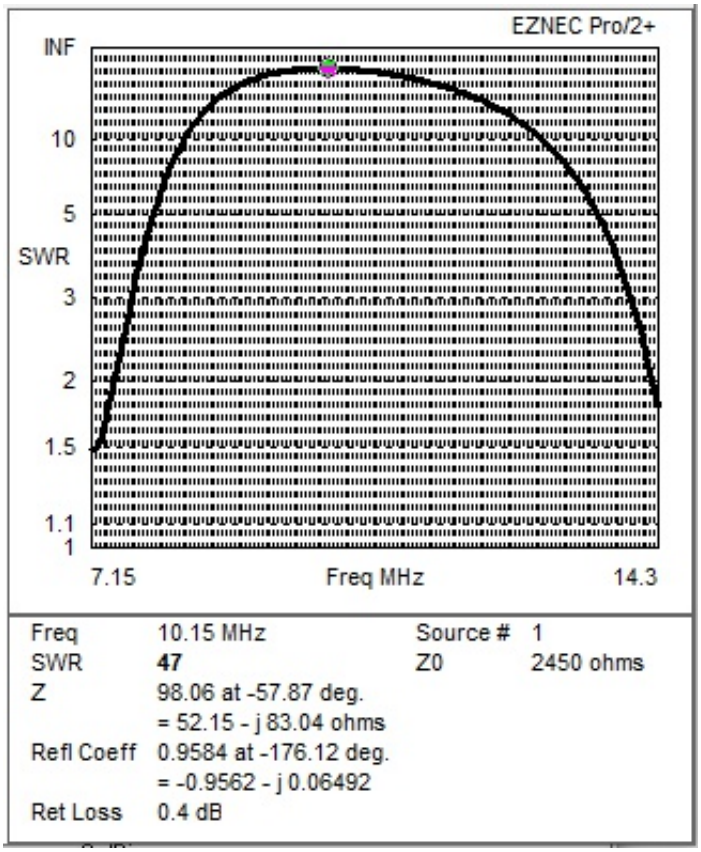
antenne zelf: alles. Een loop antenne is een resonante antenne. Als hij in resonantie is, is de stroom door de antenne maximaal en dat zorgt voor een goede afstraling. Er kunnen dan tientallen ampères lopen in de antenne! Zodra je naast de frequentie zit, loopt de antennestroom enorm terug. Als je dan met de tuner de SWR 1:1 gaat zetten, gaat echt de antennestroom niet toenemen. Wat wél gebeurt is dat je het hele systeem aan gaat passen aan de 50Ω van de zender. Het gevolg is dat de kabel gaat stralen, maar niet dat je loop antenne goed afgestemd gaat worden. Deze fout wordt wel vaker gemaakt. Ook EH-antennes zijn resonante antennes en die moet je niet aanpassen met een tuner als ze niet op hun resonantiepunt gebruikt worden. Een beste kans dat je kabel dan gaat zenden met de EH-antenne als top-capaciteit. Niet doen. Als je je frequentie verandert, is de enige manier om de boel weer goed te krijgen, de loop opnieuw afstemmen. Dan functioneert de loop optimaal.

Nou we het er toch over hebben: weet je waar zo'n soort fout ook vaak gemaakt wordt? Met de End-Fed antenne. Dat is Engels voor eind-gevoed, dus met een D. Niet End-Fet; er zit geen transistor aan het eind. De End-Fed is een antenne die elektrisch gezien voor de golflengte waar hij gebruikt wordt, een veelvoud is van een halve golf. En waarom: Aan het uiteinde van een draad kan geen stroom lopen, alleen maar spanning staan, zie het plaatje rechtsboven. Daarom is de impedantie aan het uiteinde van de antenne erg hoog, en moet er dus getransformeerd worden. Doorgaans gebeurt dat met een 1:49 impedantietransformator, dus met een wikkelverhouding van 1:7. De 50Ω wordt dan opgetransformeerd naar bijna 2,5kΩ waardoor de draad goed aanpast aan de set. Dat geldt dus voor een draadlengte van 1/2 golflengte of veelvoud daarvan. Een van de voordelen van de End-Fed antenne die geroemd wordt door amateurs is dat er geen tegencapaciteit nodig zou zijn. Maar dat is niet waar. Ook deze antenne moet zich ergens tegen afzetten. Zonder verdere maatregelen dient de buiten-mantel van de coax als tegencapaciteit. In het



Stroom/spanningsverdeling in een End-Fed antenne

ideale geval is de tegencapaciteit een kwart golflengte, waardoor de impedantie aan de kant van de antenne laag is en de meeste stroom in de tegencapaciteit zal vloeien. Omdat de impedantie in het voedingspunt zo hoog is, mag de tegencapaciteit ook een stuk korter zijn dan een kwart golflengte. Maar hoe hoger dien-tengevolge de impedantie van de tegencapaci-teit, hoe meer stroom er in de buitenmantel van de coax zal vloeien. Gebruik je dus geen extra tegencapaciteit, dan zal de buitenmantel van de coax als zodanig dienstdoen. Kan je dan niet een mantelstroomsmoorspoel gebruiken om stroom op de buitenmantel tegen te gaan? Jawel, maar dan wel aan de kant van de set. Want als je de smoorspoel aan de kant van de antenne plaatst, kan de terugkerende antenne-stroom nergens heen en werkt je antenne niet meer. Opa spreekt uit ervaring. Maar ik dwaal af. Zoals ik zei, werkt deze antenne goed op de banden waar de elektrische antennelengte een halve golflengte of veelvoud daarvan is. Ik zeg expres elektrische lengte, omdat deze anders kan zijn dan de fysieke lengte door toepassing van verlengspoelen. Maar nou hoor ik wel eens dat een End-Fed voor 40m het ook prima doet op 30m met de tuner ertussen. En dan springen mij toch even de batterijen uit de Philips oorbellen. Kijk even naar het volgende plaatje:



Ik heb een halve golflengte draad voor 40m even door Eznec gehaald en de SWR uit laten rekenen voor de frequenties van 7.15 - 14.3 MHz, voor een Z_0 van 2450Ω. Je ziet dat de SWR bij 7 MHz mooi laag is, evenals bij 14.3 MHz. De meeste sets vinden alles onder een SWR van 1:2 prima. Deze draad doet het dus uitstekend voor 40m en 20m. Ik heb de marker op 10.15 MHz gezet, de 30m band dus. Je ziet dat de impedantie dan 98,06Ω is. De SWR is daar nu 1:47 omdat er nog een impedantie transformator tussen zit, en deze transformeert de 98Ω een factor 49 omlaag, dus naar 2Ω! Eigenlijk zou je deze antenne zonder transformator aan moeten sluiten, maar die zit er nou eenmaal. Dus ga je met je tuner een SWR van 1:47 weg zitten regelen. Dat geeft een enorm verlies in de kabel, om nog maar niet te spreken van verliezen in andere delen van de antenne zoals de transformator. Een bijkomend probleem is dat deze lage impedantie ook vereist dat er een fatsoenlijke tegencapaciteit aanwezig is, omdat er nu wél veel stroom loopt in het voedingspunt. Dat is nu immers laagohmig. En bij een ontbrekende tegencapaciteit gaat die stroom in de coax lopen, met allerlei instralingsproblemen tot gevolg. Kortom, als je werkelijk je End-Fed op andere banden wilt gebruiken dan waar hij voor bedoeld is, dan kan je beter de transformator eruit laten en een tegencapaciteit toevoegen. Met bij voorkeur een automatische tuner direct aan de antenne. Zo doen ze het op schepen ook. Maar een End-Fed tunen op banden waar hij niet voor bedoeld is, is vragen om problemen. Natuurlijk krijg je met een goede tuner de SWR wel goed. En natuurlijk zal je er verbindingen mee kunnen maken: een antenne maken die niet straalt is echt heel erg moeilijk. Maar je ziet aan de antennesimulatie dat het een groot probleem geeft met de SWR en dus met verliezen en instraling. Kortom: een resonante antenne gaan tunen is geen goed idee, ook al geeft de SWR-meter 1:1 aan. Een tuner is eigenlijk bedoeld voor een niet-resonante antenne (dipool van willekeurige lengte), gevoed met een open voedingslijn (kippenladder). Een open voedingslijn heeft minimale verliezen, ook bij zeer hoge SWR. Maar een hoge SWR op een coaxkabel proberen weg te tunen is vragen om problemen. Duidelijk?" besloot Opa. Pim knikte. "Nooit geweten. Ik dacht altijd dat als de SWR-meter 1:1 aangaf, mijn antennesituatie ideaal was. Maar het is me duidelijk dat een SWR van 1:1 niet betekent dat de antenne het goed doet. Alleen maar dat de set 50Ω ziet. Ik zal de tuner er tussenuit halen en voortaan maar even naar de loop lopen om 'm opnieuw af te stemmen als ik de frequentie teveel wijzig", zei Pim. Opa knikte instemmend. "Resonante antennes tunen is een no-go. Als ze een (te) hoge SWR hebben, moet je ze weer resonant maken, maar niet tunen. Hoge SWR op coax geeft altijd veel verlies. Je gaat je yagi voor 50MHz ook niet op 70MHz gebruiken met een tuner ertussen. Dat snapt dan weer iedereen. Maar een loop antenne tunen als de SWR slechter wordt, daar staat men niet stil bij de gevolgen. Hetzelfde geldt natuurlijk bij andere resonante antennes. Ik kan je natuurlijk wel helpen met een methode om een magnetische loop op afstand af te stemmen zodat je niet te moe wordt", grijnsde Opa. "Dat lijkt me een leuk project om samen aan te werken", bevestigde Pim, terwijl hij de tuner losmaakte van de kabel.

WRTC Contest

Henny Kuyper PA3HK

Pfffffff…… eindelijk is die contest afgelopen…….

Het **World Radiosport Team Championship** is een wedstrijd op een zeer hoog niveau. Deze wedstrijd wordt elke 4 jaar georganiseerd in een bepaald land. Eerdere WRTC's zijn gehouden in Seattle (1990), San Francisco (1996), Slovenia (2000), Finland (2002), Brazilië (2006), Rusland (2010), Boston (2014), en Duitsland (2018). Dit jaar werd de WRTC georganiseerd in Italië. In het organiserende land worden een aantal teams geselecteerd die bewezen hebben in contesten te kunnen excelleren op het aller hoogste niveau. Aan de andere kant kunnen zendamateurs uit alle landen verbindingen maken met deze teams in de modes CW, SSB RTTY en FT8. De contest liep van 00:00 UTC op1 jan tot 11:59 op 10 juli 2022.

Ik had al eens eerder met succes meegedaan in een soortgelijke contest met een looptijd van één maand. Een Spaanse contest "Muerte de Cervantes" waar ik in de platina categorie eindigde. Ik was nieuwsgierig hoe de WRTC contest verliep en waar ik zou eindigen na een maand. Omdat CW mijn voorkeur heeft, zou ik alleen in die mode meedoen.

Ik heb het geweten en kon de uitkomst vooraf eigenlijk wel inschatten. Ik ben namelijk nogal competitief én verslavings gevoelig……

De maand januari sloot ik boven verwachting redelijk goed af. Dagelijks kijken of er nieuwe calls op bepaalde banden te werken waren. Op de lagere banden verliepen mijn verbindingen wonderwel. Met een te korte antenne, zelfs boven verwachting op 160 mtr. Hoewel die verbindingen heel waarschijnlijk te danken waren aan de perfecte CW beheersing van de Italiaanse teams. De condities boven de 20 mtr waren niet bijzonder en ik kon daar in de eerste maanden maar weinig II*WRTC stations werken. In januari deden er wereldwijd al snel 100.000 contesters mee. Ik eindigde die maand op de 330e plaats met 76 qso's in CW, 1 qso in SSB en 1 qso in RTTY. Niet slecht vond ik zelf. Nog maar een maandje doorgaan…..

In februari begon ik al te melden dat dit echt mijn laatste maand was. Deelnemen kostte enorm veel tijd. Telkens even kijken of er nieuwe stations zijn die ik die maand nog niet had gewerkt. Telkens de XYL informeren dat ik even naar de shack ging….

In de maand maart had Frank, PA3CNO, gezien dat ik nog steeds actief was. "Ik dacht dat jij zou stoppen" vroeg Frank. "Ja dat zou ik doen maar opeens kon ik in een paar uur veel stations werken. Nu stoppen zou zonde zijn… " was mijn antwoord. Maart eindigde ook niet onverdienstelijk en in april ging ik door. Je zult het wel raden. Ik was over de helft heen. Nog maar even doorploeteren….. mei, juni en nog een paar dagen in juli……

Het viel mij op dat sommige teams na een tijdje mijn call herkenden en mij bij naam een rapport gaven. Mij beschaamd achterlatend omdat ik niet wist welke calls en namen achter II*WRTC verscholen gingen. Ook werd ik steeds bekwamer om het juiste plekje te vinden bij split verbindingen of mijn call te herkennen tussen de vele nog aanroepende stations.

Met heel veel plezier al die maanden deelgenomen aan de contest. Maar ik realiseer mij ook terdege dat ik mijn uiteindelijke positie in de contest te danken had aan de vrije tijd, gepaard gaand met pensionering en een begripvolle XYL. Ik denk dat ik dagelijks wel 10 keer 15 minuten tot een half uur besteedde aan het zoeken naar en verbinding maken met de beschikbare stations

Het definitieve resultaat is binnen. Op de wereld ranglijst ben ik uiteindelijk geëindigd op de 152e plaats. In totaal deden er 120.000 deelnemers mee waarvan de meesten in alle modes verbindingen hebben gemaakt en ik, behoudens een 3 tal uitstapjes, slechts in één mode.

PA3HK	160 m	80 m	40 m	30 m	20 m	17 m	15 m	12 m	10 m
II0WRTC	CW	CW	CW	CW	CW	CW	CW	CW	CW
II1WRTC	CW	CW	CW	CW	CW	CW	CW		CW
II2WRTC	CW	CW	CW	CW	CW	CW	CW		CW
II3WRTC	CW	CW	CW	CW	CW	CW	CW	SSB	CW
II4WRTC	CW	CW	CW	CW	CW	CW		CW	CW
II5WRTC	CW	CW	CW	CW	CW	CW	CW	CW	CW
II6WRTC	CW	CW	CW	CW	CW	CW	CW	CW	CW
II7WRTC		CW	CW	CW	CW	CW	CW		CW
II8WRTC	CW	CW	CW	CW	CW RTTY SSB	CW	CW		CW
II9WRTC		CW	CW	CW	CW	CW	CW	CW	CW
IO0WRTC		CW		CW	CW				
IO1HQ		CW	CW						
IO2HQ									
IO5HQ					CW				
IO6HQ									
IO8HQ									
IO9HQ							CW		CW
IR1WRTC	CW	CW	CW						

Van de 3458 Nederlandse deelnemers, ben ik op de 7e plaats geëindigd.

Netherlands Show

Rank	Hunters	Score	QSO	Special Call	Bands	Modes	
1	PA0GJV	7405 Award	596	18	9	4	Log check
2	PA3ARI	6580 Award	562	13	9	4	Log check
3	PI4GAC	6341 Award	481	18	9	4	Log check
4	PA7KY	5843 Award	523	18	9	4	Log check
5	PA3BQC	4841 Award	498	15	9	4	Log check
6	PA3EVZ	4639 Award	354	18	9	3	Log check
7	PA3HK	3444 Award	178	15	9	3	Log check
8	PC4L	3056 Award	302	14	9	4	Log check
9	PD2ESI	2919 Award	229	12	3	4	Log check
10	PA3DII	2904 Award	259	16	8	3	Log check

Een langdurige contest die een hoop tijd heeft gekost maar met een, voor mij, leuk resultaat. Over vier jaar is de volgende. Ik weet niet of ik dan weer mee doe ???????

Hoewel…..

73, Henny
If it works, rip it apart and find out why!!!

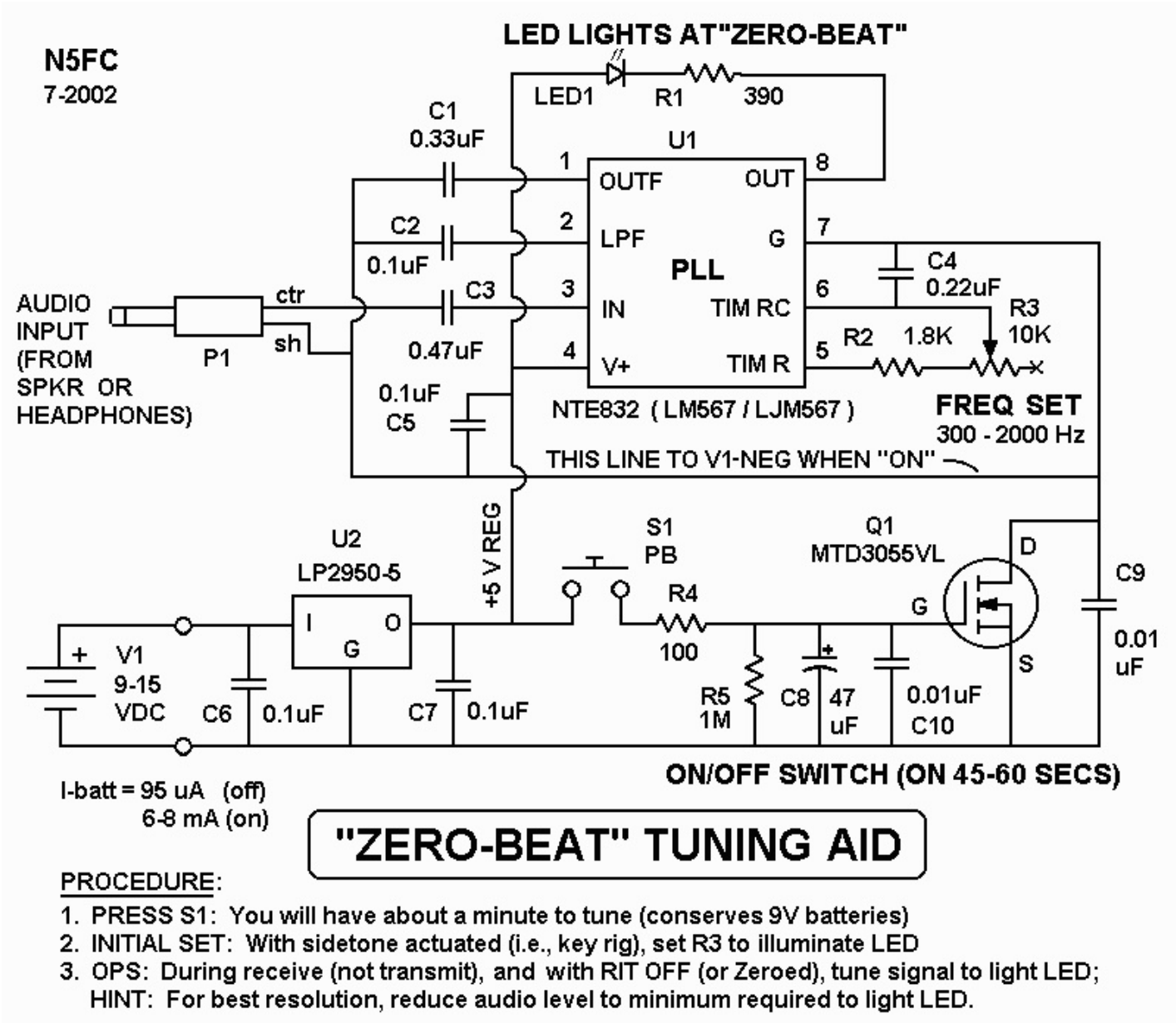
De Zero-Beat afstemhulp van Monty, N5ESE

Ik had nooit problemen met het horen van zero-beat bij het afstemmen van mijn transceiver. Maar op een gegeven moment veranderde ik in een Old Man (hey! geen OUDE man!) en kon ik de juiste toon niet vinden. Je kent het wel, typerend voor de meeste transceivers. Je bent je aan het voorbereiden op het beantwoorden van een CQ in CW, en met de RIT uit, stem je langzaam het ontvangen signaal af totdat het de magische toon bereikt die "zero-beat" betekent. Meestal is dit dezelfde toon als je CW-sidetone, 500-800 Hz, afhankelijk van je set. Je hebt een goed gehoor nodig om de toonhoogte te kunnen bepalen.

Of je nu toondoof bent, of gewoon een OM zoals ik, je zou best wel eens voordeel kunnen hebben van deze handige kleine afstemhulp. Sluit hem parallel aan op je speaker of koptelefoon. Als je klaar bent om het station dat CQ roept op zero-beat te zetten, druk je op de drukknop. Dit geeft je ongeveer een minuut om je afstemming te voltooien, waarna het apparaat zichzelf uitschakelt en daarmee de 9 Volt-batterij spaart. Terwijl hij aan is, stem je langzaam af op het signaal en stop je wanneer de LED op zijn helderst oplicht. VOILA!, Je staat zero-beat! Door het terugregelen van de audioversterking van de ontvanger wordt de detectiebandbreedte kleiner, waardoor de zero-beat nauwkeuriger is, maar ik heb ontdekt dat je gemakkelijk binnen 100 Hz kunt komen met normale volume-niveaus.

Zie het schema op de volgende bladzijde.

De schakeling is een klassieke toondetectie-toepassing van de LM567 Tone Detector/PLL. Audio van de luidspreker- of hoofdtelefoon aansluiting wordt via koppelcondensator C3 op de ingangspin van de LM567 aangesloten. Condensatoren C1 en C2 stellen de band-breedte en lusfilterkarakteristieken van de PLL in. Deze waarden zijn niet kritisch en kunnen met +/- 50% worden gevarieerd zonder merk-bare prestatieverschillen. R3, in serie met R2 en samen met C4, stelt de gewenste toon in (d.w.z. de PLL-referentie) en moet worden ingesteld op de sidetoon van de transceiver. Zender inschakelen (in een dummy load), zodat het sidetone geluid wordt toegevoerd aan de LM567-ingang, en stel R3 zo in dat de LED op



PROCEDURE:

1. PRESS S1: You will have about a minute to tune (conserves 9V batteries)
2. INITIAL SET: With sidetone actuated (i.e., key rig), set R3 to illuminate LED
3. OPS: During receive (not transmit), and with RIT OFF (or Zeroed), tune signal to light LED;
   HINT: For best resolution, reduce audio level to minimum required to light LED.

zijn helderst brandt. Met de weergegeven weerstands- en potmeterwaarden kan je de detector instellen op elke toon tussen 300 en 2000 Hz.

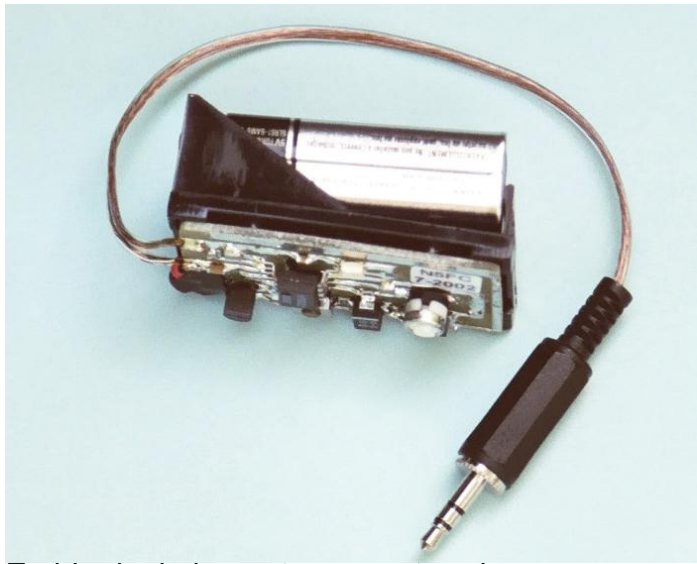
S1, R5, C8 en Q1 vormen een tijdvertraagde elektronische schakelaar voor het in- en uitschakelen van de schakeling, om de 9 Volt batterij te sparen. Zoals eerder vermeld kan je een levensduur van 6 maanden tot een jaar verwachten van een alkalinebatterij. Het werkt als volgt: U2 is een micro-power 5 Volt regelaar. Deze trekt slechts 100uA wanneer de schakeling in de "uit" of "rust" stand staat. In deze toestand wordt er geen massa aangeboden aan de LM567 en ondersteunende schakelingen. Wanneer S1 even door de operator wordt ingedrukt, wordt C8, een 47 uF-condensator, snel opgeladen en wordt de poort van Q1 tegelijkertijd boven de "AAN" -drempel van ongeveer 2,5 volt gebracht. Wanneer dit gebeurt, gaat Q1 "aan" en biedt een pad met lage weerstand van zijn drain naar zijn source, en dus bestaat er nu een massa voor de LM567 en ondersteunende schakelingen. Als de drukknop wordt losgelaten, begint C8 te ontladen via de 1M weerstand R5, maar het duurt ongeveer 45-60 seconden voordat de Gate-spanning van Q1 onder de "AAN"-drempel daalt, waardoor de schakeling weer wordt uitgeschakeld. Dit geeft de operator (en de LM567 PLL) voldoende tijd om het afregelwerk

te doen, maar zorgt ervoor dat de 9 Volt batterij gespaard blijft, omdat deze alleen significant stroom levert als dat nodig is.

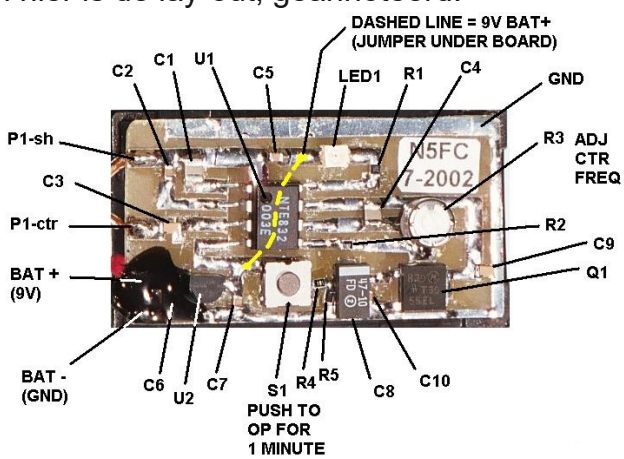
Als je de extra complexiteit van de automatische aan/uit-schakelaar niet wil, vervang je deze volledig door een conventionele SPST-schakelaar. Bewaar de regelaar echter, zodat de referentiefrequentie van het PLL-circuit niet wordt beïnvloed door spanningsvariaties als de 9V-batterij leeg raakt. Als je het automatische stroomcircuit niet gebruikt, werkt een 78L05 als spanningsregelaar uitstekend.

Ik heb de mijne geconstrueerd op een klein stukje printplaat, met behulp van een hobbymes en "cut and peel" -technieken om het pc-patroon te "etsen".

Ik heb de print zo groot gemaakt dat deze net binnen de rand van een plastic 9-volt batterijhouder past. Alle onderdelen zijn op het oppervlak gemonteerd, zelfs de "thru-hole" onderdelen (zoals draden, instelpotmeter en het DIP-IC). Waar mogelijk werden SMD condensatoren en weerstanden gebruikt. Bij montage werd de print op de achterkant van de batterijhouder gemonteerd, wat een extreem lichte en compacte montage oplevert. Op het plaatje rechtsboven is te zien hoe een en ander eruit ziet:



En hier is de lay-out, geannoteerd:



De schakeling zal ook werken met andere constructietechnieken en normale discrete componenten, dus voel je vrij om je eigen favoriete constructietechnieken te gebruiken en onderdelen die je ook in je junkbox hebt.

PA3CNO's Blog

Op het moment dat ik dit schrijf is het gelukkig geen 40 graden meer - of daaromtrent. Dat is gewoon niet mijn weer en al helemaal niet om in de shack te zitten. Om even aan te haken op Henny's artikel: ook ik heb me laten verleiden tot meedoen aan de WRTC contest, hoewel het geen contest is zoals je die normaal elk weekend aantreft op de banden met dat zenuwachtige "599 NR? NR?". Echter hoor ik nog wél tot de werkende klasse en elk kwartier naar boven rennen om te proberen weer een nieuw station te verschalken was voor mij dan ook niet weggelegd. Daarnaast beschik ik ook niet over Henny's antennepark maar over een simpele Blokker snijplank met twee draadjes eraan als Inverted-V en een zelfgemaakte open voedingslijn. Dat moest ik dus compenseren door meer modes te gebruiken en me niet - zoals Henny - uitsluitend te concentreren op CW als mode, hoewel dat wel mijn meest gebruikte mode is. In CW zijn de fatsoensnormen nog enigszins aanwezig en kom je er tenminste nog door met ons Nederlandse bescheiden vermogen van 100W,

waar ik heel wat overgemoduleerde lineairs heb waargenomen in SSB om die puntjes maar te scoren (ook van Nederlandse stations trouwens...) Ondanks mijn "handicaps" werd ik nog nummer 473 met 1800 punten in het deelnemersveld van 119.922 wereldwijde amateurs, wat mij een 21e plaats in de Nederlandse ranking bezorgde. Maar dan wel in 4 modes: CW, SSB, RTTY en FT8. Jawel, ik vind FT8 een verschrikkelijk onpersoonlijke mode, maar een contest is sowieso ook onpersoonlijk met alleen 59(9) nadat ze 8 keer om je call hebben moeten vragen. En ik had de puntjes gewoon nodig... Maar goed, inmidels zijn er weer een reeks AO1X?? stations in de lucht die een award opleveren als je er maar genoeg werkt, dus heb ik weer wat om op te jagen. Als de shacktemperaturen het toelaten tenminste...

Het artikel over de zero-beat afstemming vond ik wel grappig. Zo'n feature zit standaard in mijn FT857 gebouwd en dat gebruik ik best wel vaak. Want hoewel je aan de frequentie uitlezing meestal wel kan zien welke frequentie het station in kwestie bedoelde, zie je dat over het algemeen zelfgebouwde of QRP sets "ongeveer" op een rond getal zitten. De zero-beat afstemming helpt me dan om precies op het tegenstation af te stemmen: als het blauwe LEDje op mijn FT857 meeknippert met het CW signaal, zit ik zero-beat met mijn tegenstation. (Om dan vervolgens SPLIT te gaan zitten en mijn 2e VFO 100-150Hz hoger te zetten om door de pile-up te breken met een afwijkend toontje, maar dit terzijde). De mens heeft nou eenmaal een voorkeur voor mooie ronde getallen. Niemand maakt een afspraak bij een restaurant voor 18:53 toch? Altijd netjes 19:00. Zo was het leuk om in de tijd van het Nederlandse netje op 7.077 CQ te gaan roepen op 7.077.3. Steevast kreeg je dan het commentaar "Je zit naast het kanaal!". Hoezo? Heeft je set geen VFO dan? Maar ja, die ronde getallen hè?

Ik kauw nog op het artikel van vorige maand wat ging over die Tayloe kwadratuur detector. In combinatie met een Si5351 die twee LO signalen met 90 graden onderlinge faseverschuiving kan opwekken, moet het maken van een allband SSB/CW transceiver niet zo moeilijk zijn. Sterker nog: Als je de transceiver zo maakt dat deze I/Q signalen aflevert én accepteert, zou je met de juiste SDR software vrij eenvoudig alle modes moeten kunnen maken, zolang je SDR software het maar kan (de)coderen. Het lijkt me wel wat om dat uit te gaan werken naar een praktijksituatie. Ofwel: zo'n ding eens te gaan ontwikkelen. Hans Summers G0UPL belooft zo'n ding op zijn QRP-Labs site al vier jaar in de vorm van een QSX, en ik heb er al een paar keer naar gevraagd, maar ondanks dat hij bij hoog en bij laag beweert dat de ontwikkeling nog steeds doorgaat zie ik wel allemaal nieuwe dingen verschijnen zoals de QDX (voor FT8 adepten een onwaarschijnlijk mooi vakantie setje) maar die QSX komt er maar niet. Die QSX zou een allband SSB/CW QRP transceiver moeten worden die inclusief 10 banden en behuizing onder de $150 zou moeten kosten. Dat wilde ik graag hebben, maar ik trap inmiddels bijna op mijn baard van die 4 jaar wachten. Dus waarom zelf niet zoiets gebouwd. Met die Tayloe (de)modulator zou zoiets toch wel moeten kunnen. Ik denk aan een paar opties: een I/Q versie die dus in combinatie met een computer gebruikt moet worden, een volledig stand-alone versie of misschien een hybride versie die beide opties ondersteunt, afhankelijk van de complexiteit. Wat zou jouw voorkeur hebben? Ik kan wel weer eens een poll op de website zetten, dan kan je dat laten weten. Dan heb ik komende winter weer wat te doen en onze uitstapjes naar Liechtenstein zijn altijd ideale kraamkamers om dit soort ideeën uit te werken. Maar verwacht de komende tijd door de temperaturen in de shack niet teveel van me HI.

Ik pak hier ook maar meteen het afdelingsnieuws mee, want daar kunnen we kort over zijn. Afgezien van dat je boven in de 40m band een aantal van onze clubgenoten tegen kunt komen zo rond 21:30 vanaf hun vakantie adres, liggen de clubactiviteiten stil tot september. Pas dan pakken we de bijeenkomsten weer op en hopen we iedereen weer te zien.