Experimenten, die resulteren in een PSK31/QPSK ontvangstconverter

door Wim Kruyf PAoWV

In dit artikel beschrijft Wim PAoWV een aantal experimenten, welke resulteerden in een PSK31/QPSK ontvanger. Dat bij experimenten niet alles gaat zoals we het graag zouden willen zien, zal iedereen duidelijk zijn. Maar Wim liet zich ook door tegenslagen niet uit het veld slaan.

Inleiding

Tegenwoordig hoor je nauwelijks meer RTTY in Baudotcode maar PSK31 of soms QPSK.
Je kunt dat met een PC waar een geluidskaart in zit decoderen, als je daarvoor ergens een gratis programma binnenhaalt. Winpskse is nog beschikbaar, maar wordt niet meer onderhouden, dat is geschikt voor PSK31. QPSK, MixW en dergelijke hebben het stokje overgenomen.
Allemaal heel mooi, net als de kooptranceivers, maar je weet niet wat je aan het doen bent, en het genoegen om zelf iets werkends te ontwerpen en te bouwen is er dus niet meer bij.
Er is echter ook zonder PC met PSK31 en QPSK te experimenteren.

Ik beschrijf hier leerzame navolgbare experimenten die leiden tot een werkende en nabouwbare PSK31 decoder, zonder gebruik van een PC. Voor publicatie is in verband met de beschikbare ruimte een en ander ingekort, maar de essentie van het zelf al experimenterende tot een werkend apparaat te komen en de voldoening die je daaruit kunt putten, is overeind blijven staan.
Het voordeel zoals ik dat zie is dat je weet en begrijpt wat er gebeurt, en dat het allemaal in een zeepdoosje past. Wellicht kan dat allemaal in een IC'tje, en/of veel beter en efficiënter, geen idee, ik ben gewoon onder de hanebalken in de shack aan de gang gegaan en beschrijf hier de experimenten ter leeringhe ende vermaeck ende nabouw en voortborduren.

PSK31

PSK31 is fasemodulatie met twee fases die 180 graden uit elkaar liggen met een bitduur van precies 32 ms.
Makkelijk zul je denken, dan keer je gewoon abrupt de fase om, maar als je dat doet dan heb je eigenlijk een dubbel-sideband zender met suppressed carrier, die met een blokgolf wordt gemoduleerd, dus een (relatief) lekker breed spectrum. Overigens niks breder dan de oude vertrouwde RTTY FSK met de shift teruggedraaid naar 0. Om een idee te geven: je hebt dan zijbandcomponenten die bij een 1010 patroon waar je mee fasemoduleert liggen op 15 Hz vanaf de onderdrukte draaggolf (dat zijn de gewenste componenten) en dan ongewenste componenten op 45 Hz en steeds 30 Hz verder. Een zijbandniveau van 25 dB lager dan de gewenste component bereik je pas op 285 Hz van de draaggolf.
Niet alleen is dat een breder spectrum (570 Hz dus), maar dat kan ook hinderlijk zijn voor andere verbindingen vlak in de buurt, en bovendien gaat daar een deel (19%) van je zendvermogen in zitten dat je eigenlijk liever hebt zitten in de zijbandfrequentie die meetelt voor de S/N verhouding bij de smalbandige ontvanger.

RTTY kon je gewoon in klasse C uitzenden, dus niet lineair. Als je bij tweefasemodulatie echter de bandbreedte tot nauwe grenzen wil beperken, dan kun je dat niet want het omschakelen van de fase bij minimaal mogelijke bandbreedte gaat gepaard met een amplitudewijziging van het signaal. Dan heb je dus een lineaire versterker nodig.

Een tussenoplossing is om de fasemodulatie middels frequentiemodulatie met een sinusvormig signaal te bewerkstelligen. Dus bij constante amplitude niet met een ruk van de ene naar de andere fase, maar volgens een sinusvormig verloop de frequentie even opvoeren en weer laten dalen. Je loopt dan van de ene fase naar de andere langs een cirkel in plaats van langs een rechte in het fasordiagram. Klasse C eindtrap is dan OK, frequentieverdubbelaars en andere niet lineaire zaken ook, en de bandbreedte is niet minimaal maar de spectrumbreedte van -25 dB bij modulatie met een 101010 (PSK31 idle) patroon, wordt reeds gehaald op een bandbreedte van minder dan 90 Hz

Je bandbreedte is dan namelijk een stuk beperkter geworden, 99% van je zendvermogen zit dan in een band van 125 Hz volgens de betrouwbare vuistregel dat de bandbreedte van FM 2 maal de zwaai plus 6 maal de modulerende frequentie is. De zwaai is, bij 180 graden in 32 ms, 16 Hz. De modulerende frequentie is de halve baudsnelheid dus 16 Hz, zodat de bandbreedte volgens de vuistregel dan uitkomt op 128 Hz.

De sterkte van de component op 45 Hz van de draaggolf hangt af van de modulatieindex. Dus als je met je klasse C versterker toch rapporten "uitstekend, hoe krijg je dat voor elkaar, jij hebt wel een ultralineaire linear" wilt krijgen van hams die met software de derde orde intermodulatievervorming meten van een idle PSK31 signaal, dan zorg je dat je modulatieindex zo is dat dat derde orde product net 0 wordt. Dat kan met FM, J3(6,5)=0, hi. Het is wel een voorbeeld van de Wet van Behoud van Ellende, dus vraag niet naar de andere componenten die dan niet gemeten worden.

Houd je je echter aan standaard FM, die de gewenste fasedraaiing per bit geeft, dan ligt de component op 45 Hz afstand van de draaggolf al op -34 dB.

Wil je de bandbreedte verder beperken tot het minimaal mogelijke, dan komt er tevens amplitudemodulatie bij, want dan ga je in het fasordiagram van de ene naar de andere fase op dezelfde versnellende en afremmende manier maar in een rechte lijn en wordt je zender een dubbelzijband suppressed carrier zender.
Je spectrum is dan beperkt tot twee zijbandfrequenties bij een 1010 repeterend fasemodulatiepatroon, en de bandbreedte is dan dus 31 Hz in dat geval.
Op frequenties w+u en w-u liggen dan de enige overgebleven componenten van boven- en onderzijband als w de onderdrukte centraalfrequentie is en u de modulerende 15 Hz telegrafie in repeterend 1010 patroon. Nadere toelichting is te vinden in het hiervoor in CQ-PA nr 10 2007 gepubliceerd artikel over de PSK31TX.

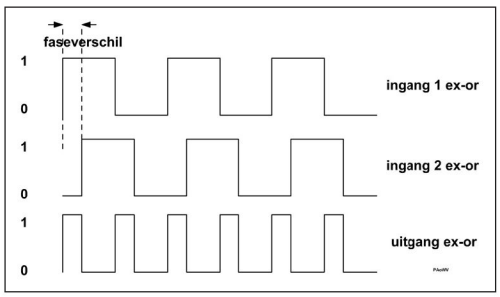
De omhullende is een halve sinus en de fase is gedurende zo'n halve sinus constant maar keert abrupt van teken om (180 graden verschil) als de amplitude het 0-punt passeert. Wel abrupt omkeren dus, maar als op dat moment de amplitude nul is, is de bijbehorende storing ook amplitude 0.

Fasedemodulator

Het is direct duidelijk dat als je fasemodulatie wilt demoduleren je een fasedemodulator nodig hebt.
Dat zou je kunnen doen met een ex-or poort 74LS86 die in mijn junkbox ligt met dank aan PAoBAT.
Dat is een simpel IC (4 stuks in een behuizing) dat een uitgang 0 geeft als beide ingangen gelijkluidend zijn, 11 of 00 dus, en anders komt er een 1 uit.

Experiment 1: Een fasedemodulator

Zet je op een ex-or poort twee in faseverschoven blokgolven van dezelfde frequentie dan komt er alleen een 1 uit gedurende

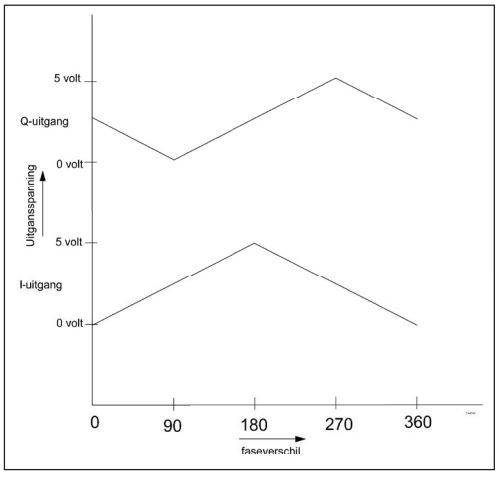


Figuur 1: Werking van een ex-or gate.

de tijd dat die verschillen en de rest van de tijd een 0. Zie figuur 1.

Laat je de faseverschuiving tussen de twee aangeboden symmetrische blokgolven van dezelfde frequentie groeien van 0 tot 360 graden, dan komt er een blokgolf uit de ex-or die tussen 0 en 180 graden faseverschil van de ingangen, steeds hogere markspaceverhouding heeft tot op precies 180 graden alleen een mark aanwezig is, en boven 180 graden weer daalt om 0 te bereiken bij 360 graden, dan is alleen een space aanwezig.

Neem je de gelijkstroomcomponent van die blok op de uitgang, dan is die evenredig met het faseverschil tussen 0 en 180 graden.
De gelijkstroomcomponent oftewel de markspaceverhouding van de blok op de uitgang van de ex-or staat in fig. 2 onderste deel van de tekening. De uitgangsspanning van een RC lid achter de ex-or poort op de y-as is daar uitgezet tegen het faseverschil tussen de twee blokgolvige inputs op de x-as.



Figuur 2: Uitgangsspanning na RC lid = gelijkstroomcomponent.

Neem je een tweede ex-or, poort 2 te noemen, waarop het binnenkomende signaal ook wordt aangeboden, maar waarvan je lokale bloksignaal 90 graden verschoven is t.o.v. het lokale signaal op de andere (poort 1), dan krijg je een gelijkstroomcomponent van het uitgaande bloksignaal op de uitgang van poort 2, die eveneens in figuur 2 getoond is bovenin de tekening.

Haal je de gelijkstroomcomponent uit de blokken op de uitgangen van poort 1 en poort 2 met een RC netwerkje en zet je de uitgangen daarvan op de X en Y platen van een scope in de XY mode, dus de uitgang van poort 1 op de X-as en de uitgang van poort 2 op de Y-as, dan doorloopt de stip op de scope een vierkant met een hoekpunt omlaag, als de faseverschuiving van het inkomende signaal van 0 tot 360 graden doorloopt t.o.v. het lokale signaal op een van beide poorten. De hoekpunten van het vierkant zijn links X=0 Y=2,5 volt; rechts X=5 Y=2,5 en dan boven X=2,5, Y=5 en onder: X=2,5 en Y=0 volt. Je kunt dus het faseverschil op de scope aflezen, en dit is dan een vectorscopefunctie voor digitale signalen. Je ziet op de scope immers het fasordiagram, waarbij de amplitude constant blijft.

Is het frequentieverschil tussen de lokale oscillator en het binnenkomende signaal bijvoorbeeld 40 Hz, dan wordt die faseveschuiving van 0 tot 360 graden 40 keer per seconde doorlopen en komt er dus een vierkant op de scopebuis te staan, dat 40 keer per seconde op de buis geschreven wordt.
Ga je het frequentieverschil opvoeren, dan krimpt het vierkant in elkaar omdat de RC leden die hogere frequentieverschillen van de gelijkstroomcomponenten beginnen af te knijpen.

Om twee 90 graden verschoven blokken op pakweg 1 kHz als lokale signalen te kunnen maken wordt uitgegaan van een symmetrische blok op de dubbele frequentie en een stel D-flops, de ene schakelt om op de opflank en de andere op de neerflank van de 2 kHz symmetrische blok. Dat kan met een inverter en daar gebruiken we een overgebleven poort van de 74LS86 voor in

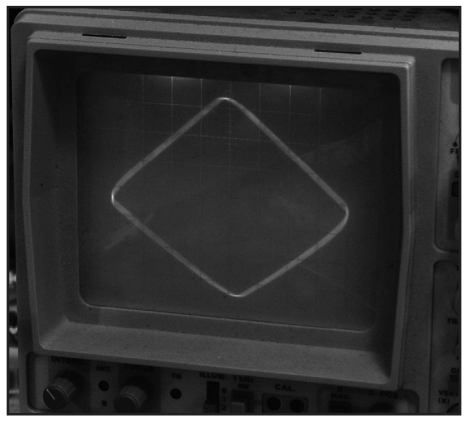
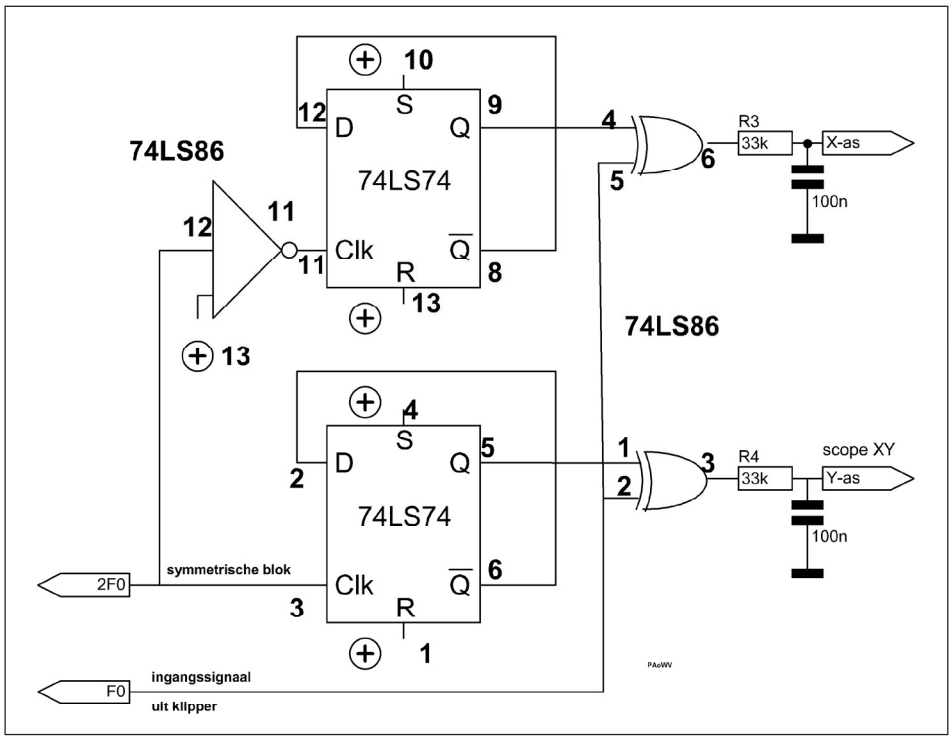


Foto 1: Vectorscope display met 10 Hz verschillende inputfrequenties.

de proefopstelling. Twee exemplaren van de in CQ-PA nog verschijnende DDS synthesizers die ik bouwde komen voor dit experiment goed van pas als signaalbronnen.
Figuur 3 geeft het schakelschema voor deze proef.

Experiment 2

Je kunt vervolgens een PSK31 signaal erin stoppen, maar dat gaat dan eerst door een limiter LM339 om er TTL signalen van te maken.
De fase is dan hetzelfde maar de amplitudevariatie is er uit. Daar wordt het dus breedbandiger van, maar dat zal ons worst wezen, want het signaal zit in de schakeling en niet meer in de ether.
Vervolgens bied ik een PSK31 signaal aan en dan toont de oscillograaf van de vectorscope een stel stippen die diametraal tegen over elkaar ergens op het vierkant liggen. Dat PSK31 signaal kun je eventueel ook halen uit een PC met een PSK programma dat op zenden staat.



Figuur 3: Een digitale vectorscope.

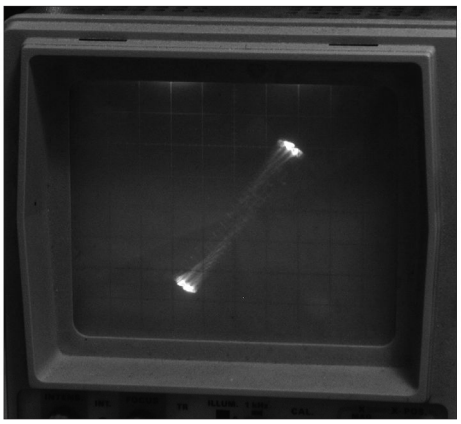


Foto 4: PSK31 op de vectorscope.

Nu we toch bezig zijn ook maar even een 4 fase QPSK signaal uit de PSK31TX gehaald, en dan liggen er 4 signaalpunten op een vierkant evenredig verdeeld langs de omtrek en de verbindingslijnen (transities van het ene naar het andere punt) zijn dan de diagonalen en de zijkanten van een vierkant.

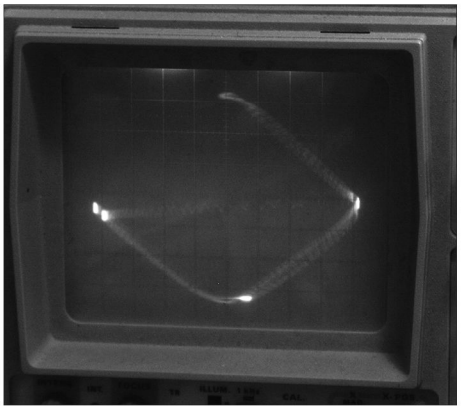


Foto 5: QPSK op de vectorscope.

Het beeld toont de waarnemer gewoon een vierkant met vier feller oplichtende hoekpunten waar langer verbleven wordt, dus heldere punten en alle verbindingslijnen diagonalen en zijden van het vierkant, die de optredende overgangen van een naar een ander punt aangeven.
Staat je lokale oscillator niet op precies de dubbele frequentie van de ontvangen PSK31, dan kruipen de twee (of vier bij QPSK) punten langzaam langs de omtrek van het op zijn punt staande vierkant op de scope.
De punten komen dan dus verder en minder ver uit elkaar te liggen.

Gebruiken we de dubbelsideband suppressed carrier methode (met een lineair) dan zijn na limiting van het signaal in de ontvanger deze beelden standaard, de fase wijzigt immers momentaan op de nuldoorgangen, dus snelle overgangen van het ene naar het andere vette punt op de scope. Zou je maximaal bandbeperkte FM gebruiken in klasse C gezonden, dan gaat de transitie van de ene naar de andere positie soepeler, dan worden de punten dus lijnstukjes op het vierkant. De doelfasepunten moet je dan dus momentaan bemonsteren op het moment dat het signaal daar mogelijk verwacht wordt. De bemonstertijden moet je dan dus korter nemen. Na limiting kun je dus nog vaststellen of van een FM of DSBSC signaal gebruik gemaakt werd, omdat het frequentieverloop anders is.

Met een analoge vectorsope, die meet voor limiting van het signaal, is dat uiteraard het geval omdat dan de cirkelomtrek of de rechte lijn worden doorlopen.

Het inzicht in dit alles groeit tijdens het experimenteren.

Laten we door een (tijdelijk) frequentieverschil van het synthesizersignaal de stippen langzaam rouleren tot ze staan op x=0 en x=5 volt, dat is een horizontale verbindingslijn tussen de twee fasepunten, dan vertegenwoordigt x het binaire gedetecteerde fasedemodulaat en heeft als uiteinde van de diagonaal van het vierkant maximale amplitude. Verloopt de lokale oscillator een kwart periode t.o.v. het ontvangen signaal, dan stuikt de X amplitude in elkaar en zit het demodulaat in de Y amplitude die dan tot volle wasdom is gekomen en dan staat er een verticale verbindingslijn tussen de fasepunten op de vectorscope.

Experiment 3

Volgens de code/lettertoekenningen gepubliceerd door G3PLX kunnen die bits uit de vectorscope vervolgens gedecodeerd in een microcontroller en de output als een lichtkrantje op een met de controller verbonden LCDisplay worden getoond, of op andere wijze worden weergegeven. Dat is dan kat in het bakkie.
Klinkt simpel, maar is het dat ook? Alles valt immers altijd tegen.

De controller krijgt dan dus het met de hand en de synthesizer bijgeregelde binaire demodulaat aangeboden, dat op de X-platen van de scope staat, waarbij we met de hand ervoor zorgen, door de synthesizerfrequentie in stappen van 0,1 Hz te wijzigen, dat de stip op de scope zich in de buurt van de linker en rechterpunt van het vierkant bevindt. De controller zet dat TTL signaal dan om in tekst volgens de zogenaamde Varicode, die G3PLX de facto gestandaardiseerd heeft door aan alle 128 ASCII codes een bitpatroon toe te kennen.

Het controllerprogramma voor experiment 3
Het controllerprogramma dat ik geschreven heb om de vectorscope output op een LCD te krijgen als leesbare tekst, bevat een decodeertabel van 128 ASCII karakters lang. Elke entry bevat 2 bytes, omdat de varicode 10 bits per karakter kan bevatten is een byte onvoldoende. Het eerst ontvangen bit van een karakter, dat altijd 1 is, staat als meest significant in een tabel-entry. Elk laatste bit van een karakter is ook altijd 1, zodat het mogelijk lijkt een een-byte-per-letter decoderingstabel te gebruiken, door de eerste en laatste 1 die altijd aanwezig zijn weg te laten. Maar dat is schijn want dan kun je kortere codes dan 8 bits, die er ook zijn, niet onderscheiden van langere.
De gebruikte tabel had ik nog kant en klaar uit een PSK31 zenderkastje PSK31TX dat goed als signaalgenerator dienst doet voor deze PSK31 en QPSK experimenten. Die tabel gebruik ik het liefst ongewijzigd, dus het programma is daarop aangepast geschreven. Dan heb je maar een tabel nodig als je zender en ontvanger achteraf combineert in een controller.
Elk karakter wordt in de controllertabel gevolgd door twee 0 bits, omdat die al ontvangen zijn als vastgesteld kan worden dat het daarom einde van het ontvangen karakter is.
Daarachter staat weer een zelf bedachte 1 als afsluiter. Die moet er bij de ontvangen letter dus worden achtergeplakt in de twee software uartbytes. Op zichzelf kan die 1 wel belangrijk zijn, want de karaktercode wordt namelijk alvorens te decoderen eerst naar links geschoven tot de eerste 1 helemaal links staat. Als er geen eerste 1 is zou dat schuiven bijvoorbeeld eeuwig duren. De sluit 1 fungeert dus altijd in nood als eerste 1 en voorkomt dus altijd dat gedrag.

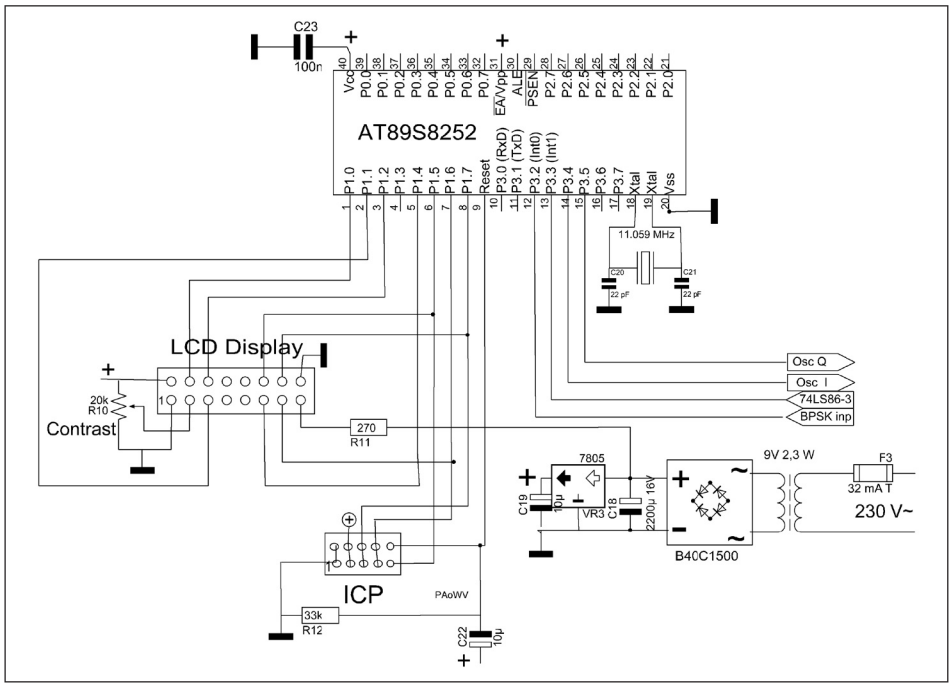
Er is een zoekroutine geschreven die van boven naar beneden door de 128 entries lange decodeertabel ploegt, en het eerste byte van een entry vergelijkt met de ontvangen code, is dat niet correct dan gelijk door naar de volgende tabelentry, is het wel correct dan wordt het tweede byte ook vergeleken. Is er een treffer dan wordt de bijbehorende ASCII code afgegeven die identiek is aan het tabel-entry nummer, zo niet dan is, als de tabel geheel is doorlopen, de carry set. Dan is er dus een niet aan een ASCII-teken toegewezen varicode ontvangen. Geprobeerd is hoe lang die routine erover doet om alles te doorzoeken, dat duurt het langst als de varicode overeenkomt met de laatste entry in de tabel. Dat blijkt 1,5 ms te zijn. Dat meten gebeurt door voor en na de zoekroutine een uitgangspootje van de controller hoog en laag te maken in het programma, dan zie je aan de lengte van de blokgolf op de scope de benodigde tijd.
Een PSK31-bit duurt 32 ms. De zoektijd is verwaarloosbaar ten opzichte van de bittijd, dus er is vooralsnog geen poging ondernomen om de tabel op te splitsen in tabellen met gelijkluidend eerste byte, wat het zoeken zou versnellen. In dat geval zou elke tabel-entry ook de bijbehorende ASCII code moeten bevatten omdat die dan niet meer overeen komt met de tabelindex.

Dan is er een interruptroutine die geac-

tiveerd wordt op de overflow van een 16 bits timer-0 in mode 1, die elke 32 ms optreedt. Die werkt dus als bitklok. De bedoeling is dat deze interrupt periodiek in het midden van een bit optreedt. Daar wordt dan gekeken of het niveau uit de vectorscope 0 of 1 is. Dat niveau wordt vergeleken met het niveau van het vorige bit dat daartoe steeds bewaard wordt. Is er een niveauwissel, dan is er een fasewissel en dus moet een 0 geschoven in de uartbytes, is er geen wissel dan moet er een 1 geschoven in die bytes. Dat laatste doet de interruptroutine niet zelf. die zet slechts de waarde van het in te schuiven bit klaar en een vlag 'valid' om aan te geven dat het bit geldig is. Het hoofdprogramma pollt die vlag genaamd 'valid', die door de interruptroutine geset wordt, als die vlag geldig is reset het hoofdprogramma die vlag en handelt de door de interruptroutine klaargezette bit af door die in de uartbytes te duwen. Blijkt het nieuwe bit 0 te zijn en het vorige bit in de uartbytes ook 0, dan is er een karakter compleet, en moet dat vertaald naar ASCII en naar de LCD worden gestuurd, omdat 00 in het signaal een letterspatie of idle is. Om te voorkomen dat bij ontvangst van een idle signaal voortdurend gekeken wordt in de opzoektabel, wordt opzoeken bij idle overgeslagen.
Er is sprake van een zogenaamde differential codering. Dat wil zeggen dat je niet kunt weten welke fase (0 of 180) 1 en welke 0 is, daarom wordt aan een fasewisseling dus 01 of 10 een ontvangen 0 toegekend en aan geen fasewisseling dus 11 of 00 een varicodebit 1 toegekend.

Om ervoor te zorgen dat de 32 ms klok van de interrupttimer inderdaad in het midden of dicht bij het midden van elk bit blijft optreden en niet langzaam wegdrift, is er een tweede extern getriggerde interruptroutine die op de downflank werkt van het door de vectorscope aangeboden gedetecteerde signaal na het RC-lid. Een flank betekent een fasewisseling dus een bitgrens in het ontvangen signaal, en die interrupt zorgt er daarom voor, het is zijn geprogrammeerde taak, dat de counter van de andere (bitklok)interruptroutine precies halverwege zijn waardebereik wordt bijgesteld, zodat die netjes in het midden van de bits in de pas blijft lopen met zijn overflowinterrupts.

De LCD is, om van het scherm lopen van zojuist ontvangen letters te voorkomen en de leesbaarheid te verbeteren, ingesteld op de tweede regel. Iedere keer als de tweede regel vol is, wipt die een regel omhoog, de oude eerste regel is dan weg, en de tweede regel begint weer van voren af aan op een gewiste tweede regel te schrijven. Dat bevalt qua leesbaarheid beter dan een zogenaamd lichtkrantje.
Het schema van deze schakeling staat in figuur 4.



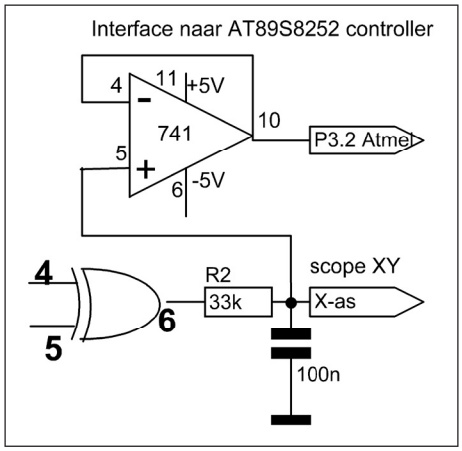
Figuur 4: Microcontroller met decoder en LCDisplay.

Test met een PSK31 signaal

Dan komt de grote proef: krijg ik tekst op de display als de vectorscope gevoed wordt met een PSK31 signaal en de uitgang ervan naar de controller gaat. Antwoord: Nee er gebeurt niks, de display blijft blank.
Na zoeken blijkt dat de vectorscope die achter zijn RC-lid een hoogohmige uitgang heeft, wat de scope prima vindt, dat die uitgangsspanning in elkaar stort omdat die geen stroom kan sinken, wat nodig is als hij een processorpoot laag wil maken. Daar heb ik met een op-amp 741 (junk box had niks beters te bieden) een voltage follower tussen gezet, wat jammer is want nu was ook een -5 volt voeding nodig. Zie figuur 5.

Nu is er wel een goed signaal op de scope dat op de processorpen komt, en het werkt!

Met vlagen werkt het, want de fase van de synthesizer is niet stil te houden. Er is verloop, dus een seconde of 8 is er leesbaar signaal en dan weer een seconde of 8 wachten, dan komt er weer een periode



Figuur 5: Een tussengevoegde voltage follower.

leesbaar signaal. De synthesizer zit er dus een dertigste hertz naast en is niet nauwkeurig genoeg instelbaar (dat is 0,1 Hz per stap) om hem stabieler te houden.

De tekst wordt gedecodeerd zolang de vectorscope aangeeft dat de fase zich in een gebied 45 graden links en rechts van de gewenste fase bevindt. De polariteit maakt niet uit, dus 180 graden verder zit ook zo'n gebied, zodat de halve tijd decodering plaatsvindt.
Nu we zover zijn, wordt het tijd eens te gaan kijken of er een in frequentieregelbare oscillator kan worden stabiel gehouden met het signaal uit de vectorscope, zodat dat niet meer draait en dus daardoor de halve tijd onbruikbaar is. Het heeft dan te lage amplitude, dan is in die periode het Y signaal overigens wel bruikbaar.

Zo'n stuurbare generator heb ik (nog) niet gebouwd op de plank staan, maar er is capaciteit over in de controller die nu de tekst maakt, dus wat let ons om die het kwadratuur (90 graden uit fase) I en Q oscillator signaal te laten maken. Mijn testapparaat kan op 2 frequenties PSK31 maken, namelijk 977 Hz en 1954 Hz, op 977 Hz kan hij tevens een QPSK (4 fase) signaal maken, dus die frequentie ga ik aanhouden.

Experiment 4: Een regelbare I en Q genereren met de controller

We gaan nu de synthesizer van tafel halen en door de microcontroller het kwadratuur oscillatorsignaal (2 bloksignalen I en Q die 90 graden faseverschil vertonen) laten maken.
Een kwadratuur blokoscillator van 977 Hz behoeft 4 maal per periode aandacht, namelijk de momenten waarop de I of de Q signalen fascomkering behoeven. Dat is dus ongeveer elke 256 microseconde, omdat 977 Hz een periodeduur heeft van

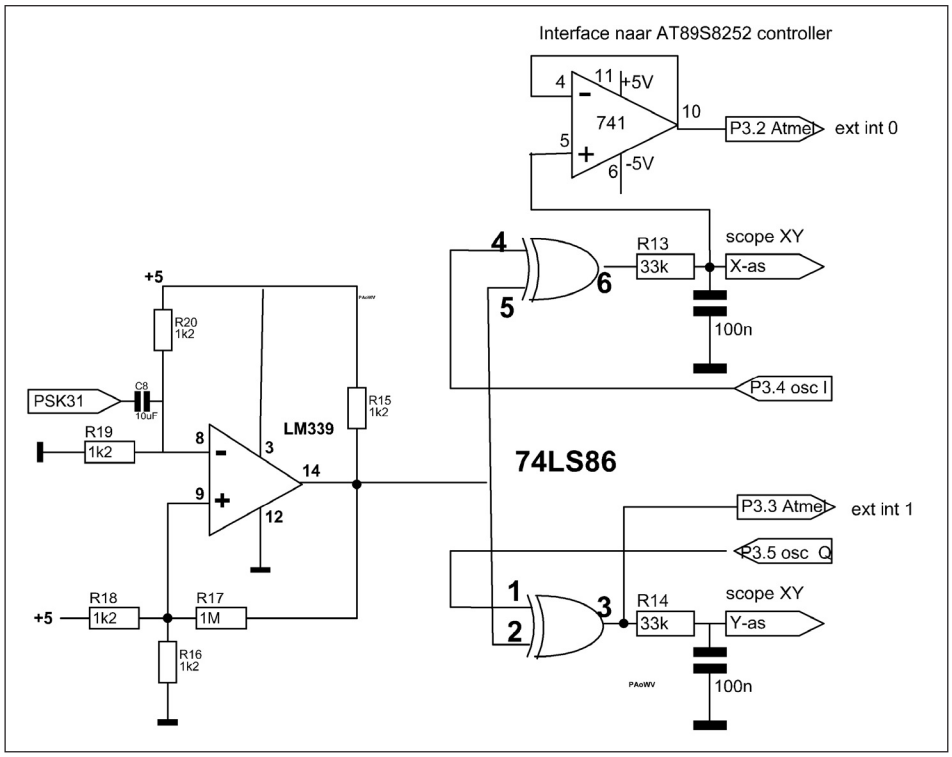
rond 1024 microseconde. Je kunt in de gebruikte controller met een nu nog ongebruikte timer_1 in mode 2, een bij overflow zichzelf op de een byte preset TH1 herladende timer gebruiken.
De interne klok van die counters is echter ongeveer 1,1 microseconde, namelijk de kristalfrequentie/12, zodat een counter in mode 2 bruikbaar is, omdat die minimaal 3600 interrupts per seconde geeft (kristalfrequentie/12 /256) en een hele blok golf I en Q dus 900 keer per seconde als minimum frequentie heeft. In de buurt gaan zitten van de laagst haalbare frequentie is niet alleen nuttig om tijd te creëren voor handelingen, maar ook omdat dan een count verschil in de preset minder invloed heeft op de frequentie. Ongeveer 0,4% (1 op 256).

We gaan een counter gebruiken in de vorm van een byte met de naam 'state', die bij elke interrupt van de I-Q oscillator een verhoogd wordt. Het laatste bit is 0 of 1, even of oneven, en dat bepaalt dus welke van de I en Q outputs gecomplementeerd moet worden; de laatste 2 bits geven aan in welk van de 4 mogelijke states de oscillator zit. Elke state beslaat 90 graden van I. Die per interrupt 1 doortellende state counter is ook bruikbaar om het aantal interrupts te tellen dat inmiddels verlopen is sinds de laatste wijziging van de counterpreset, Als je de laatste 6 bits van het state byte gebruikt heb je een 64 teller en je kunt een grensgetal nemen dat ligt tussen 0 en 64. Zodra de statecounter boven de grens komt gaat hij dan een presetcount langzamer draaien en als hij onder de grens is niet langzamer. Op die wijze kun je dan de I en Q frequentie al automatisch een factor 64 gemiddeld, let wel: gemiddeld, dichter bij de juiste frequentie brengen dan de worst case van 2,25 Hz die een vaste presetcount kan leveren. Die grenswaarde is in het programma de variabele 'fine'.
De 4 states komen overeen met de 4 zijden van het vierkant op de vectorscope.

Door het grensgetal 'fine', waarboven en beneden hij een andere preset count gebruikt, te wijzigen, kun je een regelorgaan realiseren. We hebben nu een pad waarop we voorlopig vooruit kunnen naar een doelstelling, te weten locken van de lokaal opgewekte I-Q frequentie met de frequentie van het ontvangen PSK31 of QPSK signaal.

Eerst het kwadratuursignaal maken zonder verdere poespas. Zo gebeurd. Als de state-teller, die per interrupt een verhoogd wordt, een even waarde heeft, complementeren we de ene blok en als de waarde oneven is complementeren we de andere blok, daarna wordt de state-teller verhoogd en zijn we klaar met de interruptafhandeling. Het kwadratuuroscillator signaal komt uit de processor op de pennen P3.4 en P3.5. Scope erop: ziet er keurig uit, netjes 90 graden in fase verschoven en geen jitter te zien. Je kunt de I en Q signalen ook in de XY mode op de scope zetten, dan krijg je vier stippen te zien, keurig in een vierkant.

De vectorscope is nu gewijzigd in de schakeling van figuur 6, want we hebben de externe synthesizer met de tweedelers om de kwadratuursignalen te maken, niet meer nodig.



Figuur 6: De vectorscope II na opwekking I en Q signaal.

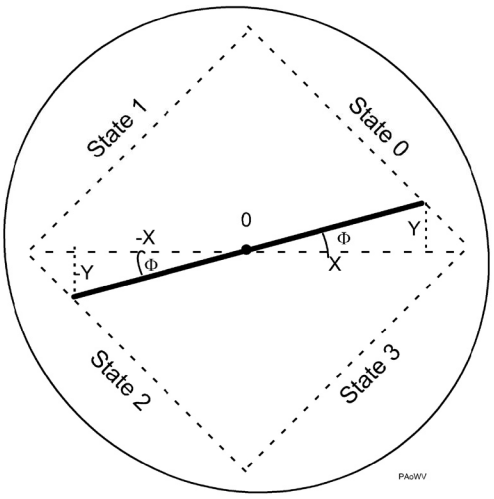
Het blijkt dat zonder verdere aanpassing de zaak toevallig al bijna in de pas loopt. Een complete omwenteling van het vectorscopebeeld met als input het kwadratuursignaal dat we met de processor genereren en het PSK31 testsignaal op 977 Hz duurt 68 seconden. De twee frequenties verschillen dus 1/68 Hz onderling. Het blijkt dat in zo'n periode 2 maal 18 seconden leesbaar schrift op het LCD scherm komt. Daaruit is te concluderen dat de faseafwijking van de lokaal opgewekte kwadratuurdraaggolf 50 graden naar links en rechts (totaal bijna 100 graden) mag zijn alvorens er geen detectie meer plaatsvindt. Dat komt dan natuurlijk omdat de X-component die de processor via de voltagefollower voedt, zover in amplitude is gedaald dat die beneden de logische detectiewaarden van de processor is gedaald.

Experiment 5:
De grote stap voorwaarts

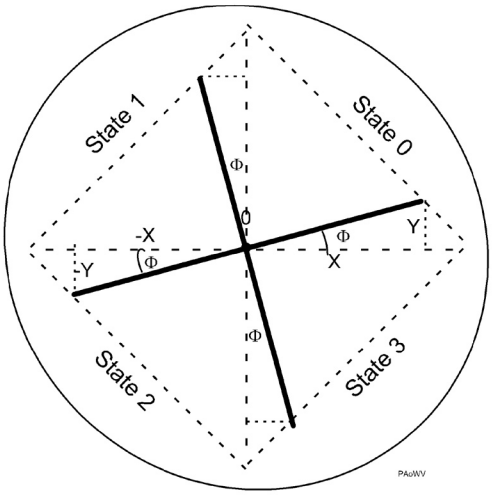
De volgende stap eist dat je kunt vaststellen of de fase van de lokale oscillator te hoog of te laag is. Tot op dit moment gebeurde dat op het oog door op het vectorscopebeeld te kijken, maar dat moet zonder oog en tevens automatisch bijgeregeld worden in een regellus.
Een idee is om de tweede beschikbare externe interrupt te gebruiken om een signaal te geven wanneer de Y van de vectorscope op zijn ex-or, dus nog voor het RC-lid, een downflank heeft. Dat kan gerefereerd worden aan de tijden van de IQ oscillator die daarop kan worden bijgesteld. Naarmate het verschil groter is, kan de bijstelling in grovere stappen gebeuren.

Voordeel: Er wordt een RC lid met interface naar de processor overbodig, dat bovendien sterk vertraagt en dat is in regellussen een ongewenste complexiteit die tot genereren van de regellus leidt, terwijl die 741 tevens een negatief voeding vereist. De volgende stap zou kunnen zijn om ook het signaal, dat we willen detecteren zonder RC lid en µA741 voltage follower, aan te bieden aan de processor, omdat we daar ook naar de momentane signaalniveaus van de blokken kunnen kijken uit de ex-or nog voor het RC lid. Dan zie je aan de horizon opdoemen dat je ook die ex-or poorten in de software kunt implementeren zodat er extern dan alleen maar een limiter voor het analoge PSK31 signaal overblijft.
Nog wildere ideeën zijn dan ook nog om de vectorscope te implementeren door de X en Y waarde voor de display op de ongebruikte poorten P0 en P2 naar buiten te sturen. Het kan kennelijk niet wild genoeg zijn, want dat wordt allemaal in dit experiment gerealiseerd.

Detectie zonder RC leden met de processor
Als je op de display kijkt van de vectorscope, dan krijg je een draaiende of in het gunstigste geval een bijna stilstaande lijn te zien, zoals figuur 7 aangeeft.
Daaruit moet de fase gehaald worden om-



Figuur 7: Vectorscopebeeld PSK31 fasehoek Φ.



Figuur 8: Vectorscopebeeld QSK fasehoek Φ.

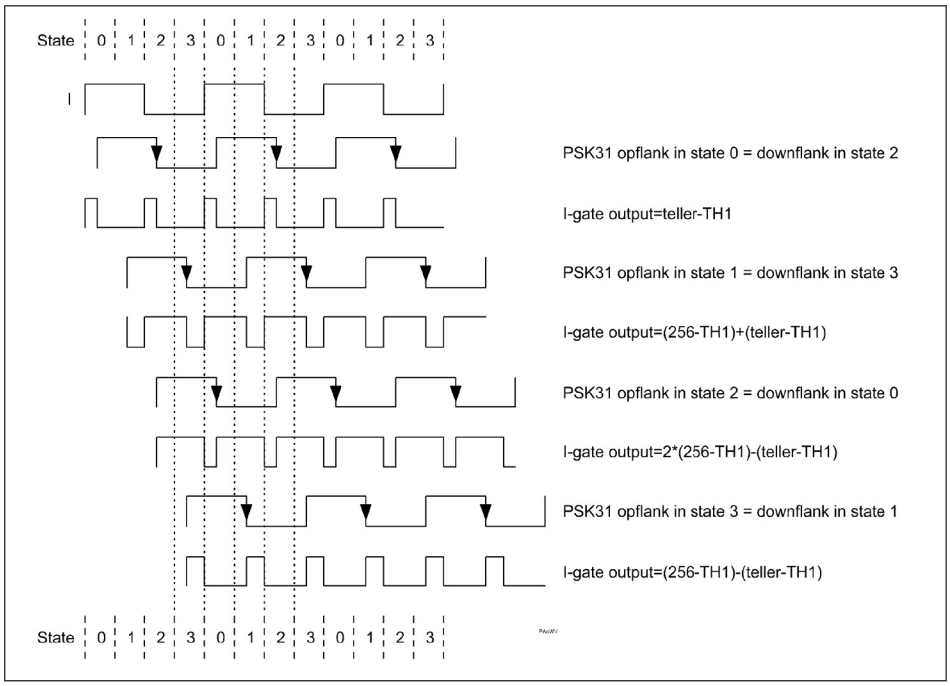
dat de fase stabiel moet worden gehouden in een regellus, de frequentie moet namelijk voortdurend bijgeregeld zodat de lijn stil of nagenoeg stil blijft staan onder een vaste hoek Φ. X is dan het zuivere gedetecteerde signaal met een bitduur van 32 ms. Er zijn echter tijdens ontvangst van het signaal 2 fases, die bij PSK31 180 graden uit elkaar liggen en in het geval van QPSK zelfs 4 fases die 90 graden uit elkaar liggen. In fig. 8 is de fasehoek Φ aangegeven op een tekening van het vectorscope scherm. Je houdt de fase gelijk en dus de plaat niet roterend, als je ervoor zorgt dat de verhouding van Y/X van de detectoroutputs constant blijft, omdat bij omkering van de fase X negatief wordt en tegelijkertijd Y negatief, zodat het quotient Y/X gelijk blijft en als regelsignaal kan worden gebruikt. De lusversterking in de te maken regellus dient voor een lineair systeem onafhankelijk van de grootte van fasehoek Φ te zijn.
Dat krijg je voor elkaar, als je regel'spanning' in de vorm van een getal, die je uit Φ afleidt evenredig is met de grootte van Φ. Dat is het geval als je uit de verhouding Y/X de Φ zelf bepaalt en dat is de functie arctan(Y/X).
Zoals uit de tekening blijkt is Y/X zowel voor de ene signaaltoestand als voor de 180 graden verschoven andere signaaltoestand beide dezelfde Φ. Je hebt Y/X of -Y/-X in die gevallen.
In geval van QPSK vind je de hoek in de twee extra gevallen als arctan (X/Y) en welke formule je moet toepassen wordt bepaald met de state die aangeeft op welke vierkantszijde van de vectorscope de fase ligt.
Wat ons dus te doen staat, is de analoge output Y van de detector en de analoge output X, zoals die na de RC netwerken en voltage followers beschikbaar is, om te zetten in digitale getallen, die aan de processor aan te bieden, die ze moet delen en de arctan nemen om de hoek Φ te vinden die evenredig is met de gewenste numerieke sturing van de reeds gerealiseerde in frequentie wijzigbare kwadratuur oscillator. Dat eist dan twee AD converters en ik voel geen klap voor die complicatie.

Ik sla dus een stap over en ga kijken of ik niet de hele RC leden en ADC's kan overslaan en direct de digitale output van de ex-ors van de vectorscope in de processor kan verwerken.

De digitale output van de vectorscope zijn in pulsbreedte veranderende signalen, die geen µA741 voltagefollower en RC lid meer nodig hebben, dat kan dus alvast weggesloopt worden. Die 977 Hz output X en Y van de ex-or gates kunnen, als je de mark space verhouding bepaalt, elk een getal opleveren dat evenredig is met de gelijkstroomcomponent.

We moeten dus de markspace verhouding gaan bepalen van dat 977 Hz signaal uit de I en de Q ex-or poorten.De processor is geschikt om een interrupt te geven op een externe downflank, niet op een omhooggaande flank. De vraag is of we daarmee uitkunnen, en dat vraagt een zorgvuldige analyse.

In figuur 9 en figuur 10 heb ik voor de downflank van een periode van het 977 Hz PSK31 signaal, dat voor alle mogelijke (8) gevallen geanalyseerd.
Figuur 9 behandelt de output van de I gestuurde ex-or gate en figuur 10 de Q-gestuurde ex-or gate. Beide voor de vier gevallen dat de downflank van het PSK31 signaal in een van de vier states valt van de kwadratuuroscillator.
Het PSK31 signaal heeft zijn downflank in die tekeningen in elk van de 4 states. De bijbehorende ex-or output is getekend, en daarvan is de mark spaceverhouding, zoals die uit een aangesloten ex-or poort zou komen, in een uit de tekening afgeleide formule bij berekend.
De tellerstand van de IQ interrupt wordt daarin gebruikt om precies het moment te bepalen waarop de downflank van het PSK31 signaal optreedt. Die teller begint altijd op presetwaarde TH1 bij aanvang van een state en eindigt op 255, waarna hij weer op TH1 begint en een IQ interrupt geeft. Die counts liggen 1,1 microseconde uit elkaar (processorkristal/12), zodat er pakweg 240 posities in een state kunnen worden vastgesteld. Bij een externe interrupt van de PSK31 downflank wordt de tellerstand van de kwadratuurcounter vastgesteld, en aan de hand daarvan kan in de externe interrupt t.g.v. de downflank van het PSK31 signaal vastgesteld worden met die formules wat de markspaceverhouding van zowel X als Y is voor elk van de vier mogelijke states waarin de downflank van het PSK31 signaal optrad.
Bij een PSK31 signaal zal elk volgende downflank altijd 180 graden of 0 graden verschillen, en worden twee formules dus verder niet gebruikt. Heb je echter een



Figuur 9: Digital I fasedetectie met mark/space teller.

QPSK signaal dan kunnen faseverschuivingen optreden van veelvouden van 90 graden, zodat dan alle 4 aan de hand van figuur 9 en 10 gevonden formules nodig zijn.

Heel verrassend is het om op te merken dat je slechts een externe interrupt nodig hebt voor het PSK31 signaal zelf, en dat de IQ oscillator zijn state en de tellerstand TL1 daarvan op het moment van de psk downflank op de input, dan bepaalt wat zowel de X markspace als de Y mark space verhoudingen zijn van de niet meer nodige ex-or outputs. Die ex-ors hoeven niet eens gerealiseerd in de software, die zijn gewoon denkbeeldig geworden, hun gedrag ligt namelijk vast in de afgeleide formules in de tekeningen van figuur 9 en figuur 10, het gaat puur om de state van de IQ oscillator waarin de externe PSK31 interrupt optreedt en het tijdstip in die state dat met de interrupttellerstand TL1 van de kwadratuuroscillator bekend is, bijna op de microseconde nauwkeurig. De IQ oscillator hoeft dus niet eens meer output te leveren voor de ex-or gates, en ook de blokgolven hoeven niet meer gemaakt te worden, want het statenummer 0 tot 3 volstaat. Die IQinterrupt blijft dus nodig voor het ophogen van de state en het regelen van de finetuning.

Nu willen we wel graag een vectorscope kunnen raadplegen, en daarom wordt de met de afgeleide formules berekende X en Y binair naar buiten gevoerd op port P0 en P2, die tot op dit punt ongebruikt waren. Als je dan een analoge scope wilt aansturen op de X en Y platen kan er een analoog signaal gemaakt worden met een R-2R netwerk, waarvan ik de werking eerder in het eerder geschreven artikel over een DDS synthesizer heb uitgelegd. Kost

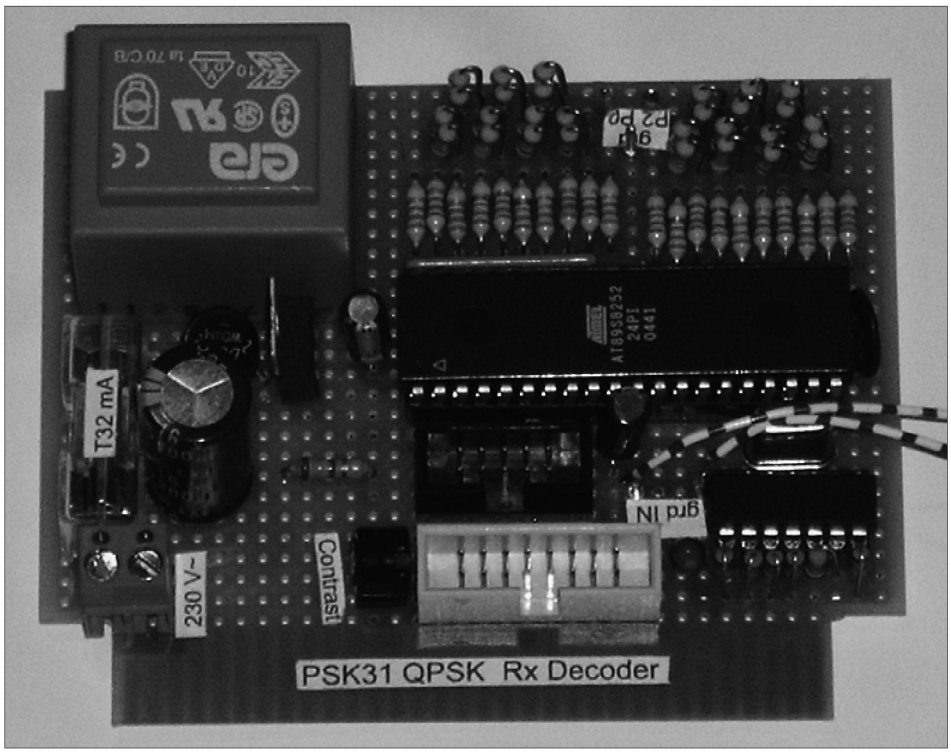
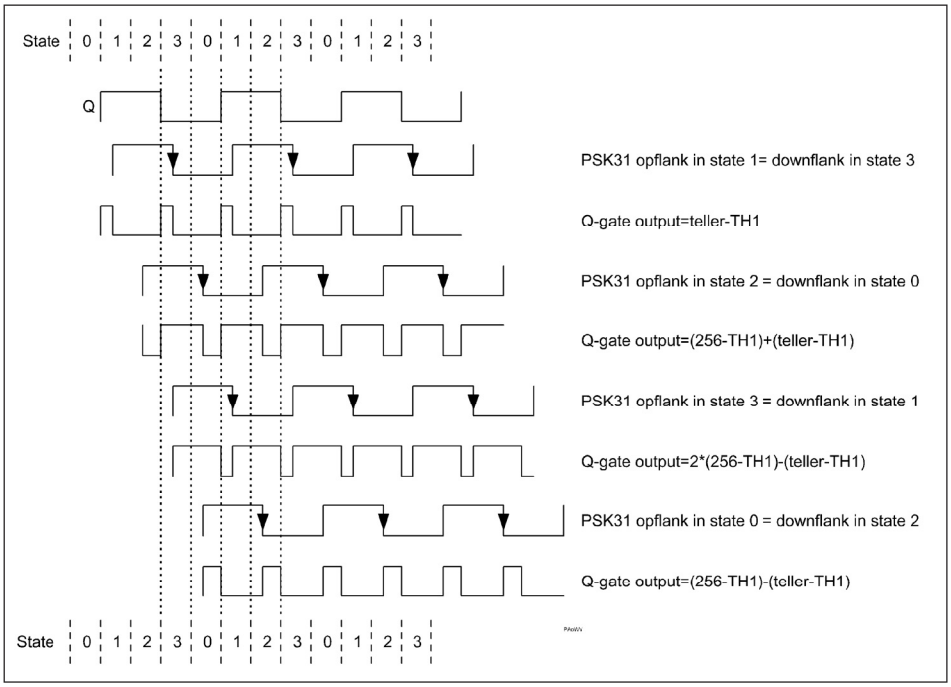


Foto 8: De print met alles erop.

per R-2R 24 weerstanden van 33K. De foto toont het bordje met de processor en een kale limiter voor het PSK31 signaal zoals het eruit ziet in het huidige stadium van ontwikkeling. De rest is weggesloopt, ex-or gates, 741 voltage follower 74LS74, een negatiefvoeding voor de 741, de aansluitingen van de IQ uitgangsblokgolven van de controller: allemaal weg. Eenvoud is kenmerk van het ware.

De externe interrupt1 wordt geactiveerd op de downflank van het inkomende PSK31 signaal, die interrupt haalt dan de state en de tellerstand op zoals die door de IQ interrupt worden bijgehouden, en al-



Figuur 10: Digital Q fasedetectie met mark/space teller.

dus is precies het tijdstip bekend in het IQ patroon waar de flank optreedt. Afhankelijk van de state worden met de gevonden formules daaruit de X en Y bepaald dat zijn de mark-space verhoudingen die de weggesloopte ex-or gates zouden hebben afgegeven aan de RC-leden. Die I en Q waarden worden via de R-2R netwerken analoog gemaakt en aan de scope in de XY mode afgegeven, zodat we weer het vertrouwde vectorscopediagram krijgen te zien, maar nu gegenereerd door de processor.

Eerst zijn de R-2R netwerken getest, door een van 0-255 doorlopende teller aan beide poorten P0 en P2 aan te bieden en de uitgang van de R-2R netwerken dan op de scope te bekijken.
De P0 port voldoet niet aan de ideale eis dat de inwendige weerstand van de spanningsbron 0 moet zijn voor het R-2R netwerk, want die heeft geen interne totempaal, of pull up weerstanden, die zijn extern aangebracht als 1 K naar +5. De Ri van de spanningsbron kan dus 1K zijn, maar de weerstanden van het R-2R netwerk zijn daarmede vergeleken groot zodat we met goede hoop de test ingaan.
Het blijkt inderdaad dat P0 een mooi rechte zaagtand geeft voor ons doel, om die als vectorscope waarde te gebruiken, en dat P2 er vier pitten in heeft zitten. Geen punt van zorg, want de precieze plaats op de scope op de mm nauwkeurig speelt geen rol.
De haakjes in een van beide treden op als een stand xx111111 overgaat in x1000000, dat mag maar een niveau schelen, maar als er een teveel van de nominale waarde afwijkende weerstand inzit kan dat fout gaan. De frequentie van de zaagtanden

blijkt 720 Hz te zijn, dat is dus voldoende representatief voor het vectorscopegebruik. Verder geen aandacht aan besteed, als zijnde van ondergeschikt belang.

Je wilt graag snel, dus met 1 byte werken, oftewel getallen tussen 0 en 255. Dat gaat niet zondermeer, want de formules voor X en Y kunnen getallen tussen 0 en 512 leveren. Het is echter mogelijk om de waarden door 2 te delen dan zit je in het getalbereik van 0 tot 256.

Krijg je QPSK binnen dan heb je 4 fases, en het zal duidelijk zijn dat je er slechts 2 kunt gebruiken voor de frequentieregeling, omdat die anders steeds gaat proberen de zaak naar de 90 gradenpunten te trekken. In eerste instantie denk je dan dat je alleen signalen van interrupts die in twee states optreden moet gebruiken voor de frequentiecorrectie. De twee andere states niet, maar je kunt de twee andere states de X en Y verwisselen, dat geeft dan een fasedraaiing van 90 graden zodat die voor de frequentiecorrectie in de pas komt met de andere twee fases.
Als je het fasordiagram in fig. 9 bekijkt van QPSK dan zal dat direct duidelijk kunnen zijn. Bij elk van de 4 ontvangen fases van een QPSK signaal is de de verhouding Y/X in state 0 en 2 of X/Y in state 1 en 3 gelijk.
Die waarden kunnen we dus voor elk van de vier fases gebruiken in de regellus die de centraalfrequentie van de zender volgt. De verhouding is bij elke state (teller)/(statebreedte-teller), bij benadering volstaat dus om de tellerwaarde van de IQ teller gelijk te houden op de downflanks van het PSK31/QPSK signaal, in welke state dat ook optreedt. De afstand van onze I en Q flanken zal dan een vast bedrag blijven t.o.v. de flanken van het inkomende QPSK signaal voor elk van de vier mogelijke fasen.

De waarde van X en Y staat in de tekeningen vermeld, zoals die volgt uit het moment van interrupt door de downflank van het PSK31 binaire signaal. Die interrupts treden 977 keer per seconde op. Afhankelijk gedurende welke state ze optreden wordt de lengte van de mark berekend uit de tellerstand TL1 van de IQ interrupttimer op het moment van interrupt die in de formules 'teller' wordt genoemd, en de state gedurende welke de interrupt optreedt. Een hoop gereken dat simpel geprogrammeerd kan worden, omdat het er steeds op neerkomt dat de IQ oscillatorflanken een vaste afstand houden met de inkomende signaalflanken.

In de PSK31 downflank-interrupt worden de waarden van Y/X in principe bepaald met een deling, dat is een ondersteunde machine-instructie in de processor voor kleine een byte getallen waarvan hier sprake is.
Natuurlijk jittert die hoek, voor de bepaling van het gedetecteerde niveau kunnen we daarom gedurende een bittijd van 32 ms, 32 stuks gevonden waarden het lopend gemiddelde van nemen.
Het lopend gemiddelde van n getallen is som/n. Voor n nemen we 32 want delen door 32 is binair makkelijk, dan laat je gewoon de laatste 5 bits weg, net zo makkelijk als delen door 100000 in ons vertrouwde decimale talstelsel. Komt er een getal m bij dan moet je dus m/n erbij doen en het oudste getal o/n eraf trekken. Je hoeft dus niet steeds alle 32 fase getallen op te tellen en vervolgens te delen. Als het gemiddelde snel verloopt, is dat een bitflank (fasewissel) van het ontvangen signaal. Een gelegenheid om de 32 ms interrupt in de pas te trekken zoals we eerder met de analoge input uit een RC lid deden. Dit lopende gemiddelde is met tijdseigen kretologie: de output van een FIR filter met gelijke coëfficiënten.
Na dit puzzel- en denkwerk is het tijd eens te kijken of de fasehoek er als een stabiel getal uitkomt dat langzaam verloopt net als de helling van de lijn op de vectorscope.

Verdere experimenten

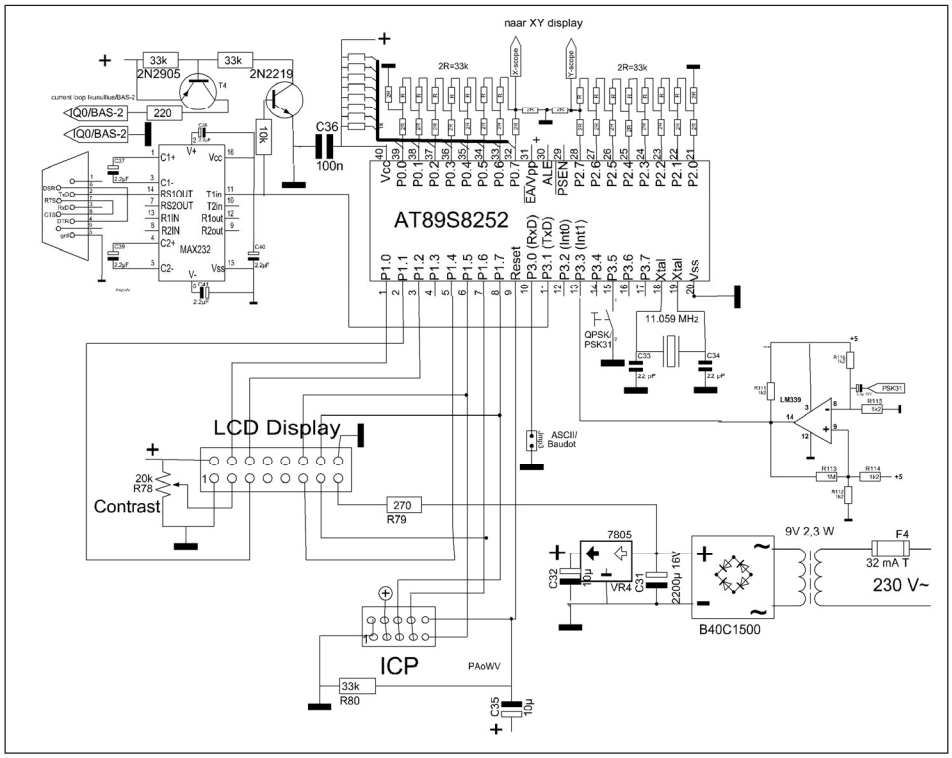
In figuur 11 is de uiteindelijke schakeling te zien van de PSK31/QPSK decoder.

De externe interrupt 1 die nu gelijk door de PSK31 uit de limiter wordt gevoed, is opnieuw geprogrammeerd volgens de in figuur 9 en 10 afgeleide formules voor I en Q output. Die leveren dus de X en Y voor de vectorscopeinput, die X en Y worden direct per downflank naar de poorten P0 en P2 gestuurd, zodat we op de scope het resultaat kunnen bewonderen en de werking van de interruptroutine en de berekeningen kunnen controleren en interpreteren.

Het is tamelijk onwaarschijnlijk dat dat allemaal in een keer foutloos werkt en daarom is het PSK signaal even vervangen door een blokgolf uit de synthesizer met nagenoeg gelijke frequentie als de IQ interrupt opwekt.

Aan de hand van de nieuwe vectorscope plaat uit de R-2R DAC's op de processor, blijkt dan in mijn geval dat op de hoekpunten van het op zijn punt staande vierkant de rondwandelende stip precies op de hoekpunten even een kwart vierkant vooruit schiet en dan weer terug.

Dat was een lastig probleem. Het blijkt dat als de externe PSK31 interrupt afhandelingsroutine bezig is, inmiddels de teller TL1 van de IQ oscillatorinterrupt verder loopt. Zit je met de PSK interrupt vlak voor een hoekpunt van het vectorscopediagram, dan passeert inmiddels die IQ teller de 255 en gaat naar een lage waarde, terwijl de state niet wordt bijgewerkt omdat die IQ interrupt afhandeling moet wachten omdat de PSK31 interrupt bezig is. In dat geval heb je dus een verkeerde doorgedraaide tellerstand te pakken, bij de oude statewaarde.
Na wat voor de hand liggend geknoei, heb ik dat definitief en grondig opgelost door gebruik te maken van een TF1 bit dat geset wordt door processorhardware bij wijze van interruptaanvraag door de IQ teller TL1, als die 255 passeert en naar zijn preset TH1 terugflipt.
In de PSK31 interruptroutine bekijk ik dat bit en als het geset is wordt de PSK31 downflank interruptroutine verder niet



Figuur 11: De uiteindelijke totaalschakeling met vectorscope en telexaansluiting.

doorlopen maar zonder berekeningen en vectordisplay afgebroken.
Dat lijkt nogal bruut, maar bedenk wel, dat dit normaal niet optreedt omdat de IQ oscillator niet wordt gesynchroniseerd dicht bij een hoekpunt, maar halverwege een zijde van het vectorscopevierkant, aangezien je dan de grootste ruismarge hebt.

Experiment 6: Faselock van de IQ oscillator

We hebben nu het voordeel dat we bij een te hoge en lage IQ frequentie niet pas na een bit van 32 ms kunnen kijken als de fase al behoorlijk gewijzigd is, maar dat kunnen we nu per periode van het PSK31 signaal reeds doen.
De fasehoek van het PSK of QPSK signaal kan voor elk van de vier states de tangens van de hoek worden berekend. Die is voor elk van de vier states de tellerwaarde gedeeld door het verschil van de statetijdsduur minus de tellerwaarde. We moeten de fase dus binnen plus en minus 45 graden constant houden. Tijdens een draaggolfperiode van het PSK31 of QPSK signaal mag de lokale oscillator er maximaal 45 graden voor of achter lopen om nog te kunnen locken zonder faseslip naar een andere state.
De PSK31 externe int routine, die voor de vectorscopedisplay zorgt, is uitgebreid met die tangensberekening in de vorm van een simpele deling. Is die tangens 1 (45 graden) of groter, dan wordt TH1 verlaagd zodat de lokale oscillator langzamer loopt en de fase dus steeds verkleind wordt, is de fase geringer dan 45 graden en de op een integer truncated tangens dus 0, dan wordt de frequentie van de lokale oscillator verhoogd, zodat de fasehoek weer klimt. Een deling om vast te stellen of die groter of kleiner dan 1 is kan zonder deelroutine door vast te stellen of de noemer groter of kleiner dan de teller is. Zo wordt alles dus steeds eenvoudiger.

Om onoverzichtelijke situaties te vermijden heb ik de mogelijke verhoging en verlaging eerst beperkt tot 8 Hz en HET WERKT! Ik kan de frequentie van de synthesizer die op de PSK31 ingang van de schakeling is aangesloten, dan verdraaien tussen 972,1 en 988,7 Hz voor de stip op de vectorscope wijzigt in een vierkant.

Ook QPSK locked zonder problemen.
Voor dit verder overdacht en geoptimaliseerd wordt willen we nu resultaten zien. De oude interrupt van timer 0 die de bittijd van 32 ms realiseerde moet gehandhaafd worden, want als de fase hetzelfde blijft, is dat een serie binnenkomende enen en het aantal wordt bepaald door de bittimer, die de tijd in stukjes van 32 ms verdeelt.
Hij zal echter niet meer op het midden van de bittijd op de ingang kijken naar de uitgang van het RC-lid, want dat is er niet meer. Ook een flank van de RC output kan niet meer gebruikt worden om met een externe interrupt int0 de bittijd te synchroniseren met de binnenkomende bitreeks en ervoor te zorgen dat de flanken in het midden van de bittijd blijven optreden.

Stel je ontvangt PSK31 en je hebt twee reeksen fasemonsters, van elk 32 bits 000011..11 en 111100..00, dan kun je onmiddellijk herkennen dat je je bittijdinterrupt 4 psk31 perioden dus 4/30 bittijd moet vertragen.
Realisatie gebeurt door in de bittijdinterrupt, die tevens nog steeds gebruikt wordt voor de vergelijking met een vorig ontvangen bit, het differential decoded nieuwe bit klaar te zetten, die pakt de lopende som aan van de in de loop van 32 ms opgezamelde fases in de PSK31 downflank interrupt, zet de som op 0, en gebruikt het resultaat om vast te stellen wat de waarde van het nieuwe bit is. Het gemiddelde van de ontvangen fases wordt gebruikt om de teller vooruit of terug te schuiven zodat de bitteller in de pas blijft lopen.

Een aantal experimenten is hier i.v.m. de ruimte weggelaten, omdat het idee hoe te werk wordt gegaan duidelijk is en er hier een werkend apparaat is verkregen.

Wie, na het lezen van dit relaas de smaak te pakken krijgt, en op eigen houtje verder wil experimenteren dan ik hier beschrijf, kan op Internet zoeken naar de Costas loop.

QPSK

In de testzender heb ik destijds QPSK gerealiseerd, dus een testsignaal is buiten een PC om beschikbaar. Ontvangst van QPSK kan dit apparaat wat de fases betreft zonder meer aan, je hebt immers 4 states. Je kunt er zonder meer 4 fases mee detecteren, dat blijkt als ik het testsignaal erop zet in QPSK mode. De vectorscope output toont de synchronisatie aan. De QPSK kan zinvol zijn, als je een Viterbi decoder toepast die fouten kan corrigeren. Die vereist 32 registers van 32 bits lang, dus 128 byte RAM, plus nog extra RAM voor het registreren van de kwaliteit van de 32 mogelijke bitstromen, om te bepalen welke aan zijn staart in aanmerking komt om aan de gebruiker, na een vertragingstijd van gemiddeld 5 letters, aan te bieden. QPSK wordt zover mij bekend veel minder gebruikt, dus daar heb ik verder niet veel werk van gemaakt.
Ik kan QPSK ontvangen als het een voldoende sterk signaal is, omdat er in mijn proefmodel nog geen foutencorrectie wordt toegepast. Dat kan zinvol zijn, want als je wilt zien wat QPSK foutendetectie voor je doet sluit je deze converter aan en tevens de converter met foutendetectie en kun je vergelijken wat het resultaat is.

QPSK zendt een uit vier fases uit voor een 1 of 0, afhankelijk van de voorafgaand uitgezonden 4 bits. De simpele decodering die ik toepas kijkt naar de laatst ontvangen 4 bits, voorspelt daaruit welke fase een 1 zal moeten zijn, door opzoeken in een 16 entries lange tabel van 16 mogelijke waarden van de vorige 4 bits, en als de ontvangen fase anders is dan verwacht, dan wordt er een 0 gedetecteerd.

QPSK eist dus een ander bitdecoderingsalgoritme, dus een andere int1 routine. Er is 1 bitinterruptroutine voor beide afhandelingen PSK31 en QPSK, maar kort na het activeren van de interrupt wordt gekeken naar een schakelaarstand of die in of uit staat, en dat bepaalt dan de vertakking naar QPSK of PSK31 bitafhandeling.
Viterbi foutcorrectie kan met de beschikbare RAM ruimte wel worden gerealiseerd voor 8, wellicht 16, in plaats van 32 bits registers.

Display

De LCD display bevalt me in de praktijk niet in de huidige vorm, er staat te weinig tekst op (16 tot 32 letters), zodat je aandacht voortdurend door die display wordt vastgehouden gedurende een QSO.

Daarom heb ik tevens een RS232 interface op de print gezet.
Daarop kun je zelfs een oude telex zetten, omdat ik Baudotcode heb geïmplementeerd, waarvoor ik een Ikunullius met BAS-2, die ik in 1975 in CQ-PA publiceerde, uit het stof heb opgegraven. Die geeft 16 regels van 64 karakters op een oude zwart-wit CCITT monitor. Niet bestaande karakters in Baudot worden dan overgeslagen. Omdat de Varicode in een wisselend lettertempo binnenkomt is er een circulaire buffer in het controller-RAM gezet, die de verschillen opvangt.

Een keuzejumper bepaalt of de tekst er in Baudot 45,45 baud of ASCII 2400 bps uitkomt.
In geval van ASCII heb je weer een mechanische teletype KSR33 of een PC nodig die hyperterminal draait. Dat laatste is een lachertje, en daarom heb ik een nieuwe schakeling met een grafische LCD 128 bij 256 punten ontwikkeld die voor een aantal zaken in gebruik is, mede in dit geval voor tekstdisplay van de PSK31rx.

Bouw en gebruik

Alles past op een half euro (10 bij 8 cm) gaatjes printplaatje. De opstelling van de onderdelen blijkt uit de foto 9.
Gebruiksgemak wordt vergroot als je seriele uitgang gebruikt om een telexmachine op te zetten of een Ikunullius die een en ander op monitor zet. Een jumper bepaalt of de output op de RS232 connector ASCII of Baudot code is.
Een en ander is dan wat comfortabeler leesbaar dan het 2 regelig LCD displaytje, dat echter zijn werk kan doen op QRP velddagen of iets dergelijks.
De PSK31TX en de PSK31RX zijn gemakkelijk te combineren in een processor door de twee programma's te combineren.
Programma en geprogrammeerde IC's zijn beschikbaar tegen onkostenvergoeding, ik ben bereikbaar op mijn call@vrza.nl.

PAoWV

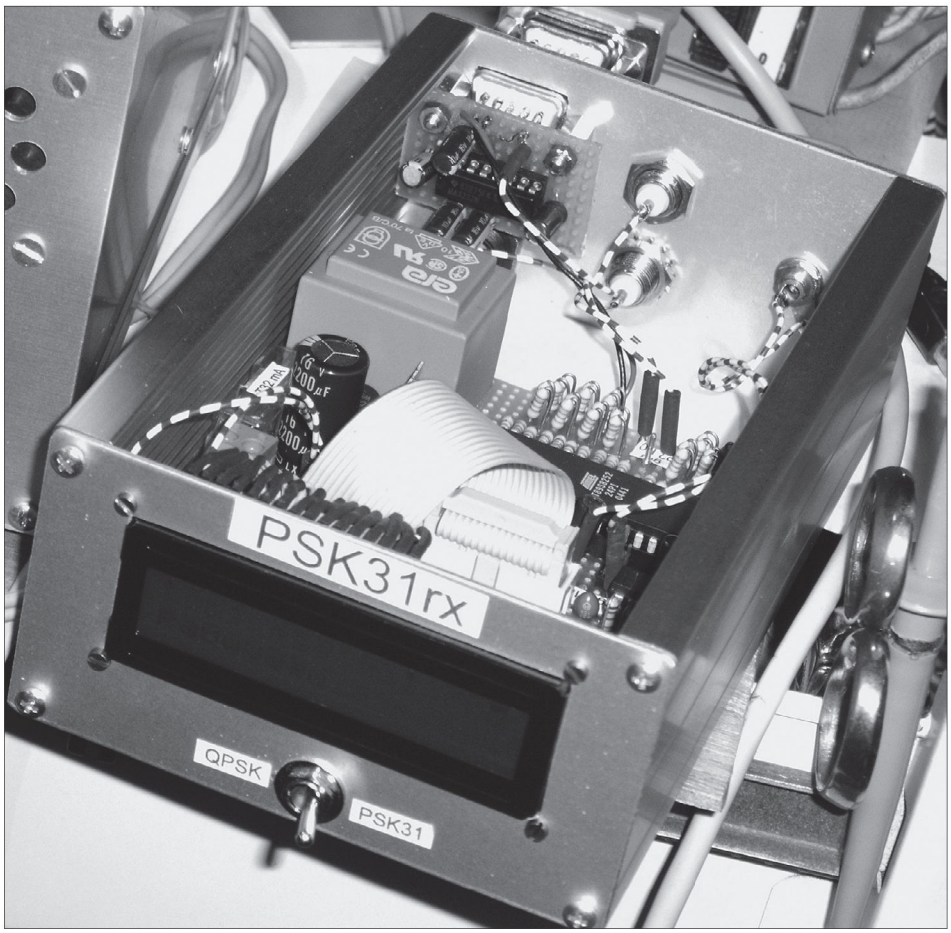
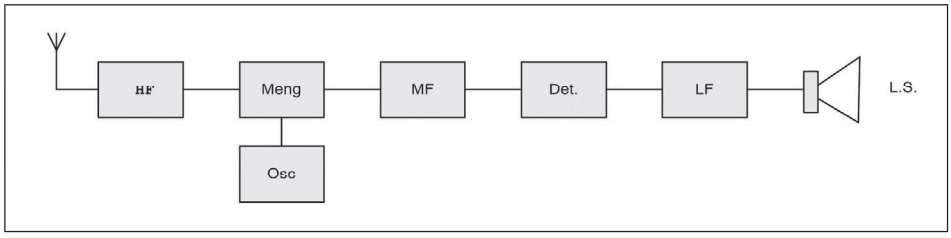


Foto 9:
De schakeling ingebouwd in een kastje.

Bespiegelingen

door George PE1APU

In de jaren '60, hadden mijn ouders geen telefoon. Zeker geen mobieltje! Als ze een telefoon gehad zouden hebben, was dit ding zeker 'Heilig' verklaard, zoals de TV en de Radio. Daar mocht ik alleen naar kijken. De apparaten werden, als mijn ouders niet thuis waren, uiteraard regelmatig op hun 'kunnen' onderzocht. De kortegolf op de radio was zeer in trek en ook in de zomer met speciale condities, de Spaanse en of Duitse zenders op de TV. De arme man moest daarna dan helemaal kanaal 4 opzoeken voor de enige zender die te ontvangen was. En dat gaf herrie in huize Reudink!
Na de 'Ambachtschool Elektrotechniek', ging ik gedreven door het feit, dat ik nog steeds geen 'elektronica-vingers' had (volgens mijn vader) naar de 'UTS Elektrotechniek'. Alle elektrische apparatuur in huis waren nog steeds 'Heilig'. Er zou daar in de rest van de jaren geen verandering in komen, maar dat wist ik toen nog niet! Inmiddels repareerde ik voor de hele buurt om een klein zakcentje bij te verdienen!
Henk (wijlen PE1ADA) en nog twee anderen die in dezelfde klas zaten, werden mijn vrienden. U begrijpt, er moest gecommuniceerd worden met hen, tijdens de studie thuis!
Ondanks het feit dat men in die tijd geen zenders, onaangemeld bij de PTT, in huis mocht hebben, had bijna ieder huishouden er minstens wel een in huis, nl. het radiotoestel!
Voorzien van een hoogfrequent deel (HF trap = voorversterker), een oscillator, mengtrap, middenfrequentversterker, detector, voorversterker en eindtrap (L.F.).
Zie schema.



Plaats de afstemming van het ene toestel ongeveer in het midden van het ontvangstbereik en stem met het andere toestel af vanaf begin bereik richting eind bereik of anders om. Dan hoort men op een gegeven moment een fluittoon in de eerste ontvanger!
Dit is de oscillator van de tweede ontvanger! Pak geen soldeerbout of schroevendraaier.
Verbindt alleen een draadje van een paar meter aan de anode c.q. collector van de oscillator en zie daar, geen fluittoon meer, maar een echte draaggolf!
Moduleer deze draaggolf en je hebt een echte 'vrienden omroep'.
Maar daarover een volgende keer.
Een ding is zeker, van te hard spelende, terroriserende, puberale middengolfradio's heb je geen last meer in de buurt. Die blaas je weg!
Kennis is macht! Of ben je nu zelf terrorist?

Groetjes
George, PE1APU