De PSK31tx, een testgenerator en zender voor digitale modi

door Wim Kruyf PAoWV

**In dit artikel beschrijft Wim PAoWV naast de testgenerator en zender voor PSK31, ook de bijbehorende theorie.
Ook heeft Wim een soortgelijk artikel geschreven over een PSK31 ontvanger. Deze zal binnen afzienbare tijd ook in CQ-PA worden gepubliceerd.**

Inleiding

Dit apparaat is in mei 2005 ontworpen en gebouwd. PAoAPA vertrok voor het slijten van zijn oude dag naar Spanje en stelde mij voor, als al lang inactief zendamateur, om een verbinding te maken met zijn nieuwe QTH.
Daar had ik wel oren naar, en wat de zendmodus dan wel zou worden?

Het zou PSK31 worden. Wat dat dan wel was, ik kreeg een artikel van G3PLX toegestuurd, en ik dacht, omdat dat artikel vrij volledig is: "O dat kan ik wel proberen te maken."
Het hier gepresenteerde apparaat is de verlate presentatie van deze boreling. Toen het apparaat klaar was, kwam ik erachter dat PSK31 normaal met de geluidskaart van de computer wordt bedreven, dat was me tijdens het ontwerp, toen ik net weer om de hoek kwam kijken in zendamateurland, onbekend.

Het werken met computerprogramma's is heel mooi, maar je weet niet wat je doet en hoe het werkt. Het zelf uitzoeken, ontwerpen beproeven en bouwen is een spannend avontuur dat leest als een jongensboek en is buitengewoon leerzaam. Naar ik merk is dat in de voorbije tijd nagenoeg verloren gegaan, omdat kopen makkelijker en goedkoper is.

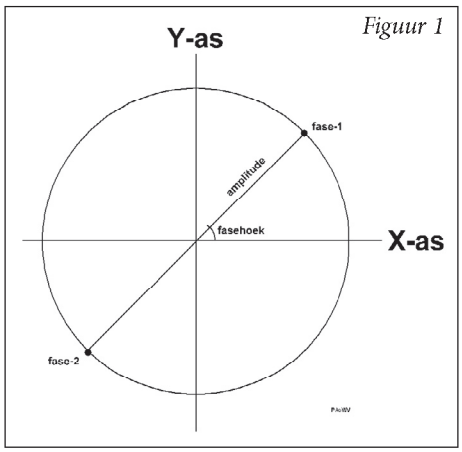
Hier wordt een PSK31 generator in hardware gepresenteerd, die ook QPSK, Hell, Morse en Baudot met 3 shifts kan genereren, zowel een testzin met je call erin evenals met een toetsenbord ingegeven tekst via een RS232 connector waar een ASCII code ingestopt kan worden met de te verzenden tekst, die in het apparaat gebufferd wordt. Met zijn recent ontworpen broertje, de PSK31rx, vormt hij een compleet hardware station.

Theorie

PSK31 is tweefasemodulatie en QPSK 4 fasemodulatie. Bij het zendexamen zit het fasordiagram in het verplichte kennispakket, maar voor de zekerheid haal ik dat hier even op.

Als je naar de propeller van een vliegtuig kijkt die draait, dan kijk je daar dwars doorheen en of die propeller nu sneller of langzamer draait, dat kun je niet zien.

Word je echter zelf vastgebonden op de propeller van een vliegtuig dat er recht tegenover staat, en de motor daarvan wordt gestart, dan draai je als een razende rond en zie je de hele wereld die verder stilstaat als vage cirkels. Draait de propeller van het vliegtuig waar je zicht op hebt even snel en in dezelfde richting als jouw propeller, dan zie je die propeller van dat vliegtuig stilstaan in de razend ronddraaiende wereld. Geeft de piloot van dat vliegtuig wat meer of minder gas, dan zie je de propeller met het snelheidsVERSCHIL met de jouwe langzaam vooruit of teruglopen.
Op die wijze kun je dat goed bestuderen. Je kunt het alleen niet navertellen.

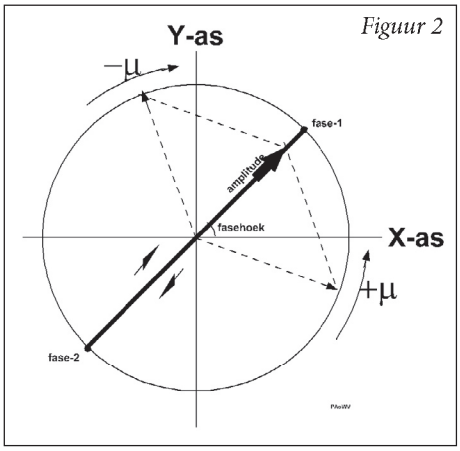


Figuur 1

Een fasordiagram (fig. 1) geeft de amplitude van de draaggolf weer, als straal vanuit de oorsprong (middelpunt van een cirkel) en de momentele fasehoek als hoek met de x-as. Bij 14 MHz draaggolf draait die lijn, die een grootte en een richting heeft en daarom vector wordt genoemd, 14 miljoen keer per seconde rond, dat is dus het equivalent van die propeller waar een stilstaande waarnemer maar kijkt.

Laat je nu echter het vlak van tekening ook 14 miljoen keer per seconde draaien (linksom), dus de waarnemer die er naar kijkt draait mee, dan staat die vector stil. Geeft de piloot gas en wordt de frequentie 1 Hz hoger dus 14,000001 MHz, dan ziet de waarnemer die vector een keer per seconde een omwenteling naar links draaien. De tekening van de vector en wat ermee gebeurt tijdens modulatie is dan een fasordiagram. Je kunt er de zijbanden intekenen die dan wat vooruit en achteruit draaien met het frequentieverschil met de draaggolf en zo inzicht verwerven over wat er precies gebeurt met omhullende en fase van de draaggolf.

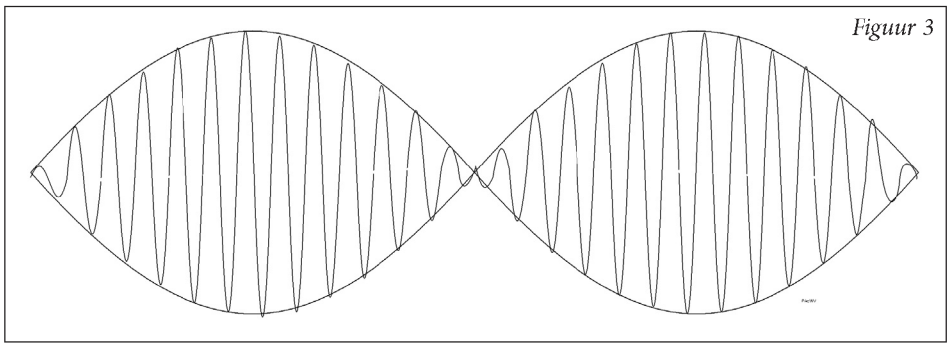
Bij PSK31, tweefasemodulatie, kan de punt van de stilstaande vector op twee plaatsen staan, die plaatsen staan dan diametraal tegenover elkaar, want de fasesprong is 180 graden en de hele cirkel is 360.
Een belangrijke conclusie is dan dat om de fase 180 graden te laten verspringen je tijdelijk, tijdens het verlopen van de fase, de draaggolffrequentie moet verhogen of verlagen. Bij verhogen loopt de vector even linksom naar zijn nieuwe positie en bij verlagen even rechtsom. Tijdelijk is de frequentie dus even hoger of lager.



Figuur 2

Het gemoduleerde signaal van PSK31 is bij minimale bandbreedte en gemoduleerd met 10101 etc. de op en neer springende vector, de punt niet langs de omtrek van de cirkel lopend, maar over de verbindingslijn van de twee fasepunten. De bandbreedte is minimaal als het verloop van het ene naar het andere punt gebeurt alsof het sinusvormig verloopt, en de modulatie is dan hetzelfde als bij dubbelzijband suppressed carrier. In het fasordiagram in fig. 2 zijn de twee zijbandfrequenties twee vectoren die tegen elkaar inlopen met de modulatiefrequentie en waarvan de resultante over een rechte lijn loopt van het ene naar het andere fasepunt. In de figuur is dat er gestippeld bijgetekend.

Zo'n signaal ziet er op een scope uit als in fig. 3 is getekend.
Hetzelfde idee als een tweetoonsignaal met een SSB zender dus. De fase keert abrupt 180 graden om op de momenten dat de amplitude van het signaal 0 is. De omhullende (momentane amplitude) van het signaal is een sinusvorm.


Figuur 3

Nu kun je niet zeggen dat fase 1 een databit 0 is en fase 2 een databit 1, want je weet bij ontvangst niet welke fase 1 en 2 is.
Daarom wordt een differential code toegepast; dat wil zeggen dat als het vorige bit verschilt met het huidige (10 of 01 dus) dan is er sprake van een 0 als databit en als er geen verschil is (00 of 11 in de ontvangen reeks bits), dan is er een 1 als databit ontvangen. Dat er geen verschil is moet niet te lang duren, want dan raak je de tel kwijt hoeveel enen er intussen langsgekomen zijn. Maar de gebruikte code voor de lettertekens, de Varicode, zorgt ervoor dat dit inderdaad nooit lang duurt.

Willen we zo'n signaal maken, dan kan dat eenvoudig met een microcontroller. Daarin gebeurt alles digitaal. We moeten dus de stap naar een analoog signaal maken. Dat heb ik gedaan door gebruik te maken van een sinustafel die voor elke gebruikte hoek de amplitude aangeeft. Table-look-up heet die techniek. Die tabelwaarde stuur ik dan als 8 bits via een outputport naar buiten waar een DAC, een Digital to Analog Converter, de waarde omzet in de analoge amplitude.

Als DAC heb ik twee uit de junk box geviste SN74167 gebruikt.
Zogenaamde BCD rate multipliers. Dat zijn IC's waar je een klok instopt en waar je (op 2 stuks) een 8 bits BCD gecodeerd getal van 0 tot 99 op kunt aanbieden. Het getal bepaalt dan hoeveel van de 100 aangeboden klokpulsen de IC's doorlaten. Die doorgelaten klokpulsen gaan een RC lid in dat dan de gelijkstroom of laagfrequentcomponent eruit haalt en dat is het gewenste signaal.
Ze bestaan ook met typenummer SN7497 als binaire rate multipliers, die kunnen per stuk tot 6 bits en per 2 dus zelfs 12 bits DAC vormen, maar die lagen niet in de junk box hier.

Is er tussen twee opeenvolgende bits van het PSK signaal geen fasedraaiing, dan gaat het signaal gewoon met volle amplitude en gelijkblijvende fase door. Is er wel fasedraaiing dan daalt de amplitude naar 0 volgens het verloop van de sinus look up table en op het 0 moment keert de fase om en groeit de amplitude weer aan volgens de sine look-up tot de volle waarde, in de PSK31 bittijd van 32 ms. Dat is het principe.

Als de twee zijbanden netjes tegen elkaar indraaien ligt de amplitude van de resultante van de twee zijbandvectoren in het fasordiagram op de rechte verbindingslijn tussen de twee fases.
De omhullende is $2a*\sin(2*\pi*f*t)$.
f : het verschil tussen de zijbandfrequentie en de onderdrukte draaggolf.
a : de zijbandamplitude.

We moeten dus een sinus maken met een vaste fase waarvan de amplitude een sinusvormige omhullende heeft en op de nuldoorgangen van de amplitude moet de fase omkeren voor het geval we van de ene naar de andere fase gaan.
Blijft de fase tussen twee buurbits gelijk dan gebeurt er niets en houden we fase en amplitude gelijk. Een blok maken met de gewenste amplitude is makkelijker, en dat is geen bezwaar, een symmetrische blok heeft alleen oneven harmonischen, de eerste ongewenste component op 3f ligt reeds 10 dB lager, het RC lid achter de rate multipliers zorgt met 6 dB per octaaf nog eens voor 9 dB, en de microfoonversterker van de zender laat 6 kHz van die component ook niet door.

Ontwerpcriteria

Ik maak gebruik van een Atmel 89S8252 processor, omdat die hier in een laatje ligt te wachten op toepassing, die, als je wilt dat de uart op standaardsnelheden kan worden bedreven een kristalklok vereist van 11,0592 MHz. Het is een kloon van de Intel 8052 met wat extra toeters en bellen, en echt snel naar de huidige maatstaven is die niet, minimaal 1 microseconde per instructie, maar wel zonder problemen soldeerbaar in de vorm van een 40 pens DIL chip. Er zit voor ons doel een 256 counter in, die zichzelf op de instelbare preset terugzet bij overflow. Die kan dus delen door 1 tot 256 naar wens van de gebruiker, en zijn klok is een twaalfde van de kristalkok, 921,6 kHz dus.

Willen we zoveel mogelijk monsters in de sine lookup table, dan moet in de bittijd van PSK31, dat is 32 ms, de amplitude naar 0 (n monsters) en weer terug naar de volle waarde (n-1 monsters) komen. Bij elke klokinterrupt van bovengenoemde teller, die optreedt als hij bij 256 komt en op de preset wordt teruggeplaatst moeten we een monster uit de tabel halen en op de uitgangspoort zetten.

De vraag is dus hoeveel monsters er passen in de halve PSK31 bittijd dat die van max naar 0 loopt. Die tijd is 16 ms. Er zijn minimaal bij een preset van nul, 3600 interrupts per seconde, We kunnen dus 3600 keer per seconde een spanning ompolen om er een blokgolf van te maken, dat geeft een blok van 1800 Hz.
Nemen we de preset hoger dan 0, dan komen er meer interrupts, en wordt de frequentie wellicht te hoog om door de audioband van de zender te kunnen lopen. Omdat de bittijd van een PSK31 signaal 32 ms is, moet in 32 ms, dus minimaal 115 keer de spanning worden omgepoold en in geval van een 0 bit de amplitude worden gewijzigd. Dan zitten er dus 60 blokgolfperioden in een bittijd. Elk blok kan zijn eigen amplitude krijgen uit een cosinustabel, en dat zijn er bij hogere frequenties meer.
Deze gegevens vormen de begrenzingen voorlopig, bij QPSK is het weer een ander en tevens aanzienlijk complexer verhaal.

Er lopen bij PSK31 en QPSK twee interruptprogramma's, namelijk timer1 overflow voor de zojuist besproken frequentie en bittijd en vervolgens draait er een uart ontvangstroutine, die via het keyboard op de RS232 ingang binnengekomen ASCII tekens kan aanpakken en met de hardware flowcontrol CTS RTS verzorgt dat geen keyboardkarakters verloren gaan door een volle buffer waar het zendgedeelte uit kan putten.

Er zijn ook andere modi mogelijk, dan hebben de interruptroutines andere functies en zijn er zelfs drie actief. Vandaar dat na het optreden van een interrupt van timer0 of timer1, een bij het opstarten gemaakt kopie van de positie van de DIL mode switches wordt bekeken en aan de hand daarvan wordt bepaald, welk van de acht vertakkingen de interrupt kiest met een indexed jump table.

Morse

In de stand Morse zorgt de snelle interrupt van timer1, die int3 in het programma activeert, dat als er morsetoon moet zijn (on=1) dat dan bij elke interrupt de portwaarden die naar de DAC gaan worden omgezet van 1 naar 99 of terug. En timer0 zorgt op int1 dan voor de morsesnelheid, die is ingesteld op 20 wpm.

Baudot

Bij baudot, 45,45 bd, staat de preset van die timer0 op een waarde die ervoor zorgt dat elke 11 ms een interrupt optreedt, dat is de halve bittijd van het 45.45 baud telexsignaal. Daarmee schep je de mogelijkheid anderhalf stopbit uit te zenden en dat is van belang om ontvangers die uit sync zijn zo snel mogelijk (weer) in sync te krijgen.

Bij een interrupt van de bittimer0 in baudotmode wordt gekeken naar een statusvariabele. Die kan de waarde 1,2 of 3 heb-

ben en wordt bij het verlaten van de interruptroutine bijgesteld.
Status 1 wil zeggen dat we in het midden van een bit zitten, de status wordt dan op 2 gezet en de interrupt verlaten. Status 2 wil zeggen dat actie geboden is.
Staat de bitcounter die tussen 0 en 7 loopt op 7, dan is er een stopbit voltooid en is het halve stopbit aan de beurt, daarom wordt de status dan op 3 gezet. Is de bitcounter in status 2 ongelijk aan 7 dan wordt er een volgend aan de beurt zijnde bit uit het te verzenden karakter gehaald en als dat mark is wordt de timer1 preset, die de toonhoogte bepaalt, van de int1 routine op de marktoon gezet en als het een 0 is op de shiftfrequentie die van de instelling van de DIL schakelaars afhangt. Tevens wordt de bitcounter verhoogd en de status dan op 1 gezet.

Er is een byte buffer tussen het hoofdprogramma dat te verzenden teken klaar zet en vertelt dat het gebeurd is door een flag1 te zetten. De interrupt leegt die buffer als de flag1 staat en reset de vlag. Is het karakter verzonden en de flag staat nog op ongeldig dan wordt overgegaan op de mark frequentie. De interrupts per 11 ms blijven gewoon doorgaan, de status is dan 3 wat wil zeggen dat er dan steeds gekeken wordt of de flag al op geldig staat.

Een karakter van 7,5 bitsduur wordt dus in 15 interrupts afgehandeld. In geval een mark wordt de preset van de toonint op een mark gezet (2125 Hz) en in geval van een space hangt het van de shiftkeuze op de DIL schakelaars af, af wat de preset is, die is dan naar keuze 24, 170 of 850 Hz hoger.

BPSK31 op 1952 Hz

De mode BPSK31, het volledige acronym van PSK31, wordt in de routine gekeken of het bit under transmission 1 of 0 is. In geval 1 wordt bij elke interrupt de volle amplitude, dus beurtelings 99 of 1 aangeboden aan de uitgangsport, afhankelijk van sign, die per interrupt van teken wordt gewisseld. Een downcounter houdt bij of het bit klaar is. Zo ja, dan wordt een volgend bit geladen en als dat er niet is een idle signaal. Op die manier wordt dus een blok gegenereerd van minimaal 1800 Hz. Is het bit under transmission een 0 dan moet een fasewisseling optreden. Dat wordt dan gedaan door bij elke interrupt de counter te bekijken en de daaraan gerelateerde tabelwaarde uit de cosinustabel te halen die de amplitude op de uitgangsport bepaalt.

Als de counter precies halverwege is, dan wordt de ompoling van het signaal eenmalig overgeslagen, dat zorgt dus voor de fasesprong van 180 graden bij de nuldoorgang van de omhullende. In fig. 10 is een tekening te zien van de flowchart van deze routine. Een flowchart (stroomdiagram) maakt het mogelijk de routine ook op andere typen controllers te realiseren.

De gekozen preset van 236 leidt op die wijze tot een draaggolffrequentie van 1952,5 Hz. Een PSK31 bit heeft een tijdsduur van 32 ms, zodat de downcounter en de tabellengte worden bepaald op 124,96 monsters per bit. Als we de noodzakelijke afrondingen maken tot 125 interrupts in een bittijd, dan wordt de transmissiesnelheid minder dan 0,03 % afwijkend van de nominale waarde.

Hell

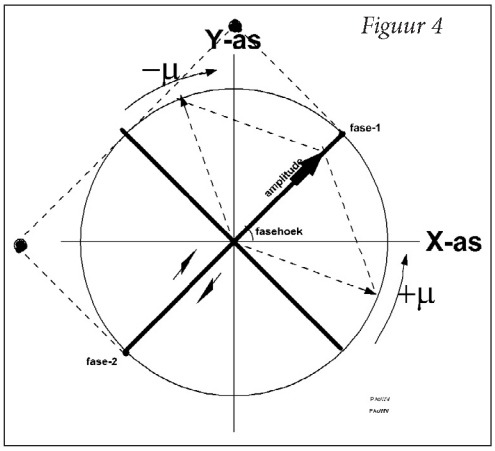
Ingeval de Hell code is gekozen met de DIL schakelaars is de gewenste snelheid 122,5 Baud en worden de letters geschreven als plaatjes van 5 bij 7 punten, in de vorm van 5 kolommen van 7 punten.
De timer1 met int0 staat op een vaste toon, en kijkt steeds of een variabele on of off is. Is hij on dan wordt per interrupt geinverteerd en is hij off niet. Dat is dus een gesleuteld signaal. De punten van het 5 bij 7 raster van een letter worden dus met on/off overgebracht net als bij morse. De timer0 interrupt op int1 die 122,5 keer per seconde optreedt, schuift per keer een bit van de 7 van een kolom dat bepaalt of het on of off wordt.
Zijn er 7 bits geweest, dan wordt een nieuw byte van die letter uit een vertaaltabel (karakter generator) geladen weer met een flag1 om aan te geven dat hij aangenomen is. Is er niets beschikbaar aan data dan worden nullen gezonden (signaal uit dus).

QPSK

Willen we ook QPSK kunnen maken en dat willen we, ook al is het net zo zinloos als destijds 16,66 toeren grammofoonplaten kunnen afspelen, dan moeten we minimaal 4 monsters per periode van de draaggolf hebben, omdat we met 4 fasemodulatie werken.

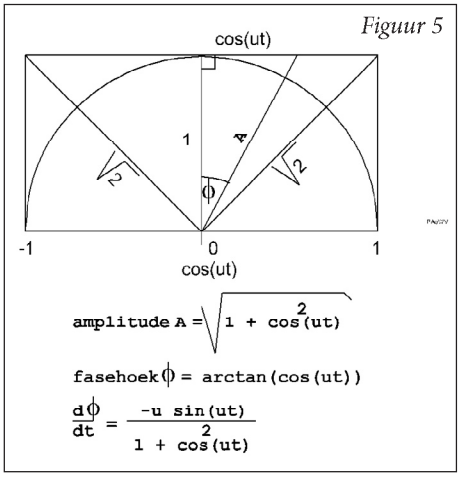
We kunnen namelijk lopen naar een fase 90 graden vooruit (hogere frequentie) naar een fase 90 graden achteruit (lagere frequentie) en tevens de al bekende 180 graden waar voor- of achteruit niet uitmaakt. En uiteraard geen fasedraaiing, dus niets wijzigen.

Nu is de vraag welk pad in het fasordiagram doorlopen wordt in frequentie en amplitude als de fase 90 graden voor- respectievelijk achteruitloopt. Dat is van belang want je eindigt wel op hetzelfde punt, maar je weet het detectiemechanisme niet.
Nu is in het artikel van G3PLX uitgelegd dat hij QPSK opvat als twee BPSK signalen, die in kwadratuur gemoduleerd zijn, dus op twee onderling 90 graden verschillende onderdrukte draaggolven.
Daaruit is het exacte verloop van de amplitude en fase als functie van de tijd te destilleren, bij 90 graden voor- of achteruitlopen van het signaal. Immers het signaal is de resultante in het vectordiagram van een van maximum naar minimum lopend BPSK signaal, over een rechte dus van fase 1 naar fase 2, en van een stilstaande vector (geen fasedraaiing) van een tweede B{SK signaal dat er loodrecht opstaat.
Dat zijn namelijk de gevallen dat de fasedraaiing van 90 graden resulteert. Fig. 4 toont het gedeeltelijke fasordiagram van QPSK als toelichting.

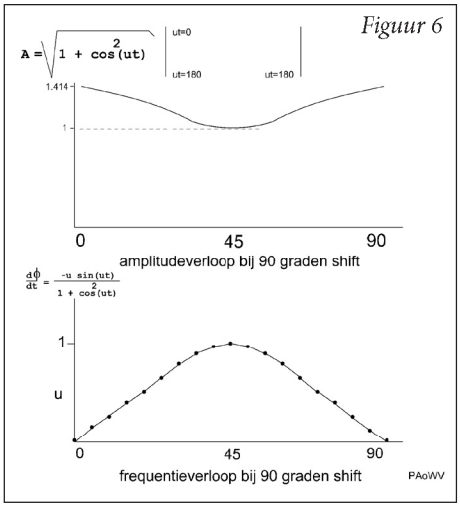


Figuur 4

Uit deze gegevens kunnen we, omdat we de lengte en fase van de draaggolfvector nu tijdens het wisselen van 90 graden exact kunnen weten, formules afleiden voor het preciese veranderen van amplitude en fase tijdens een bit dat de fase 90 graden laat wijzigen.



Figuur 5



Figuur 6

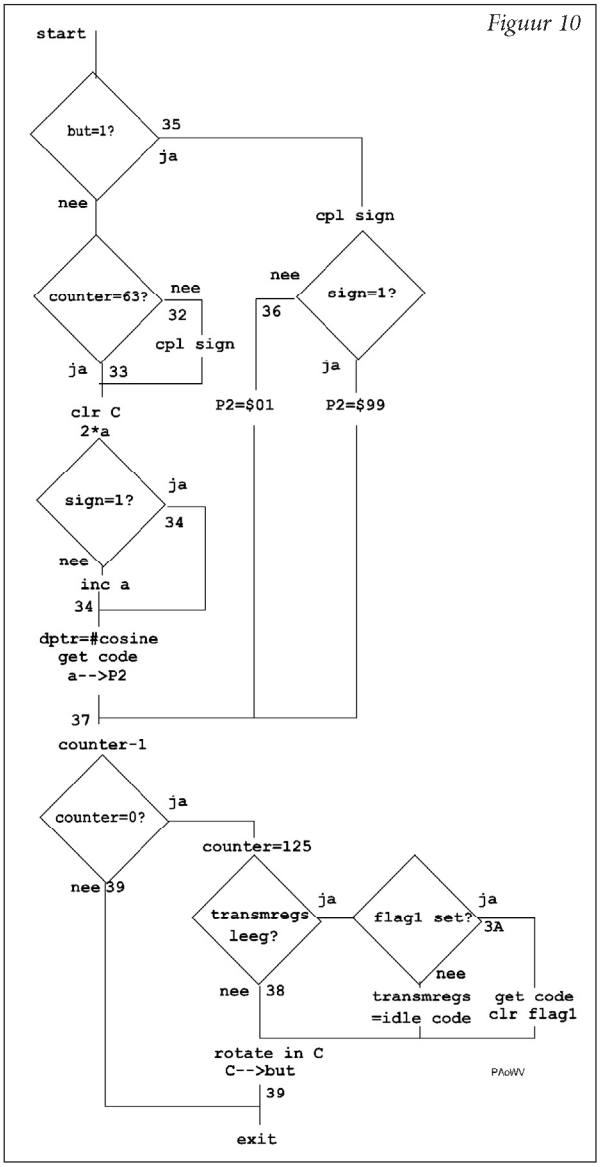
De formules staan met de tekening waaruit ze af te leiden zijn in fig. 5.

De amplitude en de momentane frequentieafwijking van de nominale draaggolffrequentie, die in deze formules staan, zijn getekend in grafiekjes in fig. 6.

Die berekende lengte en fase kunnen we weer in een tabel opnemen en wel aparte tabellen voor 90 graden vooruit, 90 graden achteruit en 180 graden fasesprong. Ingeval 90 graden vooruit is de frequentie tijdelijk hoger dus moeten we ook de preset van timer1 die de frequentie bepaalt per monster aanpassen, de tabel zorgt daar ook voor dat die gegevens geprepareerd gereed staan.
Het is natuurlijk een behoorlijke inspanning om zo'n tabel samen te stellen, dan zit je wel uren te prakken op je zakjapanner. Om dat en de inherente fouten van handberekening te voorkomen, heb ik de tabel in de vorm van een assemblerlisting laten genereren door een in C geschreven programmaatje van enkele regels.

Ook de BPSK (twee fasemodulatie dus) kan met 4 monsters per draaggolfsinus worden gerealiseerd en dat is eerst gebeurd.
De interruptroutine voor de draaggolf treedt 125 keer per bittijd op. Er zitten dan dus 31 draaggolfperioden in een bit. De interruptroutine kijkt of het bit under transmission 0 of 1 is.
De waarde 0 eist halverwege een fasewijziging en tevens een amplitudeverloop voor de verschillende perioden tijdens de bittijdsduur en een uit te zenden 1 niet. Bij 1 blijft ook de amplitude constant op de maximale waarde. Is het zendende bit 0, dan wordt gekeken of de counter die het aantal interrupts per bit telt halverwege staat.
Zo ja dan wordt de fase, die 0, 1, 2 of 3 kan zijn, modulo 4 met 2 verhoogd. Met 180 graden dus. Zo niet dan gebeurt dat niet en blijft de fase dus hetzelfde. Fase 0 en 2 zijn de nuldoorgangen van de draaggolf en fase 1 en 3 de toppen. Verder is er de normale afhandeling van het laden van een nieuw bit als de teller op 0 komt, en als dat er niet meer is van het herladen van de coderegisters en als dat niet kan van het laden van de idle code.
Het flow chart van deze routine staat in fig. 10.

Figuur 10



Interruptafhandeling van QPSK
Bij de QPSK interruptroutine flowchart is te zien in fig. 11, hebben we te maken met 4 states, namelijk geen faseverschuiving, 90 graden fase vooruit draaien in een bittijd, fase omkeren en tot slot 90 graden fase vertragen in een bittijd.
Die gewenste state wordt per bit bepaald uit de historie van 5 bits, zoals het geïmplementeerde Viterbi algoritme vereist en voldoende is toegelicht voor implementatie in het artikel van G3PLX.
Omdat we nu 4 monsters per sinus van de carrier hebben, hebben we vier fases. Nul en twee zijn de positieve en negatieve nuldoorgang, 1 is de positieve top en 3 de negatieve top van de draaggolfsinus (of blok). De routine zoekt uit wat er aan fase en state voorwaarden zijn en aan de hand daarvan wordt de uitgangsamplitude vastgesteld. Daar zijn 3 tabellen voor.
De eerste is die ook bij PSK voorkwam, namelijk amplitude als functie van de monsterwaarde, de twee andere tabellen zijn erbij gekomen. Die zijn voor de vooruitlopende en de achteruitlopende fase. In feite zijn die uit elkaar af te leiden, maar dat heb ik niet gedaan, want er is ruimte genoeg in het programmageheugen.

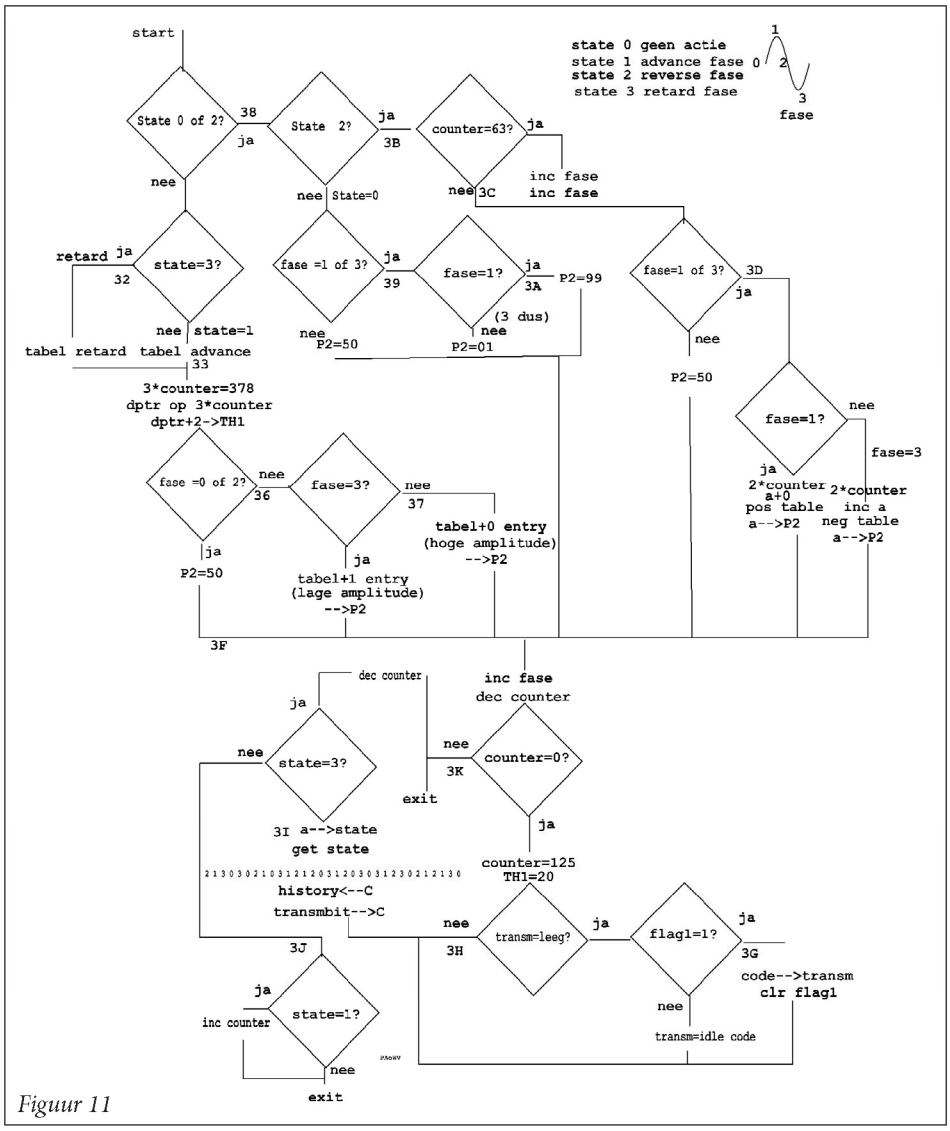
De extra tabellen bevatten voor elk monsternummer de amplitude en tevens de preset van de draaggolfcounter, want de frequentie wordt tijdens de bittijd nu ook tijdelijk in stapjes verhoogd of verlaagd en weer terug naar de nominale waarde op het einde van de bittijd. Als de frequentie verhoogd wordt gemiddeld gedurende het bit om de fase 90 graden voor te laten lopen, zou het bit bij gelijk aantal sinussen = gelijk aantal interrupts te kort duren. Daarom is de initialisatie van de teller dan een hoger en bij de vertragende frequentie wordt de teller een lager geïnitialiseerd, zodat op precies de juiste bitlengte wordt uitgekomen met de vereiste fasedraaiing ten gevolge van de tijdelijke frequentieverhoging of verlaging die resulteert in de 90 graden fasedraaiing. Dat is uiteraard een uitzoekerij, maar dat is gelukt en de resultaten zijn in de tabellen verwerkt.

Een en ander is in te zien door de volgende gedachtegang: Een bit duurt 32 ms, de timerklok van de processor loopt op een frequentie van 921600 Hz. Er zitten in een bittijd dus 29491 klokpulsen. De nominale waarde van de draaggolffrequentie treedt op bij een counterpreset van 236, dus bij 3906 Hz. Elke draaggolfperiode bevat 4 interrupts, zodat de draaggolffrequentie 976,5 Hz is. Gedurende een bitperiode van 32 ms treden dus nominaal 125 interrupts op die per 4 stuks een periode van de draaggolf geven.
Als de fase vooruit moet worden geschoven moet de frequentie tijdelijk verhoogd, maar de bittijd blijft evenveel klokperioden van de timer kosten, want die blijft 32 ms. De verhoging zal zodanig zijn dat het einde van de bittijd een kwart periode eerder wordt bereikt, dan hebben we namelijk totaal 90 graden fasedraaiing. En om de bittijd even lang te laten duren is er dan dus een extra klok nodig voor de ontbrekende kwart periode.
Verder zal de som van alle verkortingen in de presets gedurende dat bit gelijk zijn aan de totaalwaarde van de nominale preset, omdat die immers overeenkomt met een kwart van de nominale periodetijd van de draaggolf.

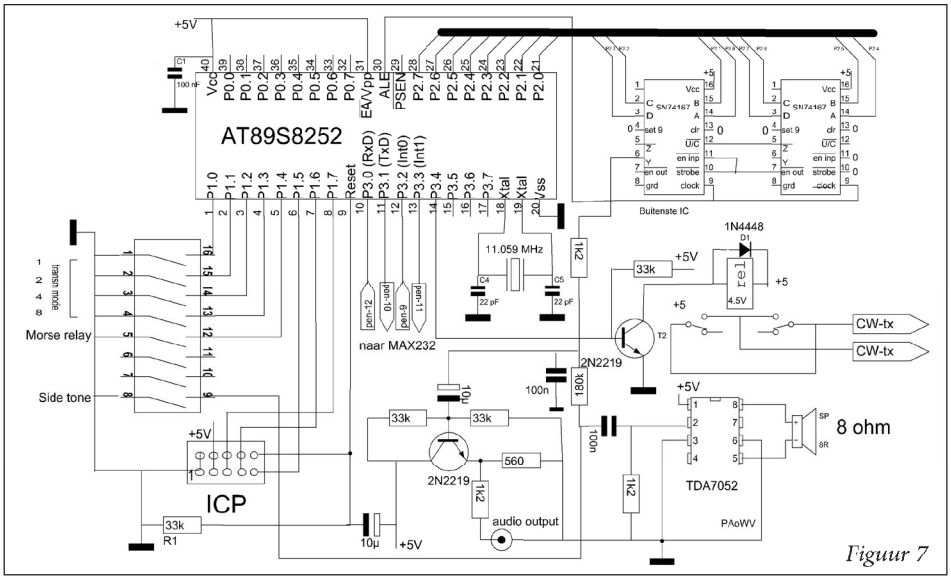
De flowchart van de interruptroutine voor QPSK is getekend in fig. 11. Die flowchart is zinvol, niet alleen om de werking in detail te begrijpen, maar vooral omdat aan de hand van een flowchart willekeurige processoren van ander type dan de door mij gebruikte programmeerbaar zijn.

Metingen

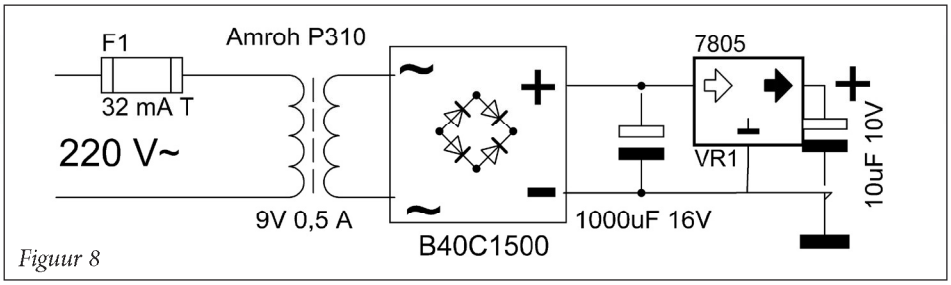
Het is van belang vast te stellen hoeveel tijd de QPSK interrupt in beslag neemt. Het is immers een stuk programma dat 3600 keer per seconde doorlopen wordt en dat qua tijdsbeslag uitschieters heeft als een nieuw karakter moet worden op-



Figuur 11



Figuur 7



Figuur 8

gehaald. Daartoe heb ik aan het begin van de interrupt als test een ongebruikte pen van de processor hoog gemaakt en aan het einde van de interrupt weer laag. De interrupt treedt periodiek op met 250 microseconde en de in beslag genomen tijd is ongeveeer 50 us met uitschieters volgens de scope tot bijna 100 us.

Nabouw

Het schema staat in fig. 7. Er is een uitschakelbaar morserelais ingezet, en een audio driver met een emittervolger achter de DAC. De ICP is de in circuit programming connector, makkelijk om de schakeling te ontwikkelen en daarna om eventueel updates van de software aan te brengen.
Een audioversterkertje met een speakertje zorgt voor de uitschakelbare sidetone. De voeding is te zien in fig. 8 en de RS232 interface staat in fig. 9 (zie pag. 000).
Er zijn geen bijzondere onderdelen gebruikt. De programmatuur is onder GNU licentievoorwaarden gratis bij me op te vragen op mijn amsat email-adres.
Op verzoek kan ik ook een geprogrammeerde processor leveren, met je call erin en met opgave van de gewenste morsesnelheid.

Conclusie

De conclusie die je hieruit kunt trekken is dat het mogelijk moet zijn met behoud van 4 monsters per periode het aantal perioden per seconde te verdubbelen. Dan komt de QPSK op bijna 2 kHz en dat is gelet op de signaalvorm prettiger omdat harmonischen dan door de microfoonversterker van de zender beter worden weggehaald.

Het is gebleken dat dit apparaat een onmisbaar hulpmiddel is geweest bij het ontwikkelen van een hardware ontvangstdecoder.

(figuur 9 zie pag. 300)

PAoWV