Een dual Si570 VFO

PAoWV

Inleiding

Dit artikel gaat over een VFO met een frequentieuitlezing op LCD, die gemaakt is middels een IC Si570. De bedoeling is, dat de VFO bruikbaar kan zijn als signaalgenerator, en het is het begin van een CW zender of CW-transceiver; voor de ware zendamateurs dus, helaas niet voor de met een uiterst moeilijk staatsexamen na langdurige studie gecertificeerde brekiebrekiebakkenisten die met een SSB koopdoos hun mening, afgedwongen door het verdrag van Rome (hij hep reg) met een antennemast van 24 meter hoog in het postzegelgrote achtertuintje van een arbeiderswijk, wensen uit te blaten over hun galstenen, voorzover die tenminste een bloedhekel hebben aan "dat gepiep", maar die wel posthuum wensen geridderd te worden met SK - silent KEY - als de hen toegemeten tijd verknoeid is met dat blaten in een microfoon. A van alfa B van Bravo E van Gelie. Allemaal niet nodig, die omhaal, als je gewoon ABE seint. Wel zo snel, doet denken aan vervlogen tijden dat voetballen nog een sport en geen beroep en oorlog was; en spaart bandbreedte. SM voor Silent Mike is een meer adequate betiteling voor het voor de langer levenden ingetreden zalige eeuwige stilzwijgen.

De techniek

Zo, die zijn weg.

Reeds enige jaren is een IC Si570 op de markt, en het IC staat in belangstelling van SDR fanaten en anderen, tevens wordt het geleverd door Funk-Amateur hun web-winkeltje op http://www.box73.de Dat winkeltje werkte in mijn geval feilloos en aldus kwam ik na betaling in het bezit van twee stuks van die SMD pootloze vliegen, twee voor het geval van verlies, je weet wel, als de xyl ze opzuigt in haar eeuwigdurende schoonmaakwoede, dan kan ik er wellicht nog een terugvinden in mijn eeuwigdurende allesbewarenwoede.

Hoe werkt dat ding ongeveer

Daar zoeken we de specificaties van op, in het onvolprezen en evenzeer zo verfoeide Internet. http://www.silabs.com/Support%20Documents/TechnicalDocs/si570.pdf Het blijkt dat er een kristaloscillator inzit van pakweg 114 MHz; wel een kristal maar op die frequentie en die afmetingen en temperatuurvariaties kan dat geen ufb zijn. Die kristalfrequentie wordt vermenigvuldigd met een 32 bits gigagetal, zodat je uitkomt ergens tussen 4,85 en 5,67 GHz, daar zit een in dat frequentiegebied verstembare VCO. Dat is met

recht Giga, daar kan zelfs je magnetronoven niet tegenop. Het is wel het klassieke PLL schema, een GHz VCO, daarop een deler en een fase comparator, met het kristal, zodat je afhankelijk van het uiteraard gehele deeltal uitkomt op veelvouden van de kristalfrequentie. (fig 1) Hier dus

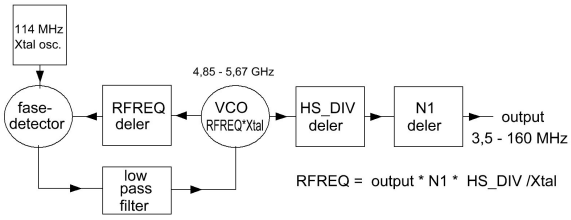


fig 1 de Phase locked loop

stappen van 114 MHz. Ga je die aan het kristal gelockte VCO dan delen om op 3,5 tot 30 MHz uit te komen dan heb je frequentiestappen van minstens 85 kHz. Dat schiet niet op; er is dus kennelijk wat bijzonders aan de hand.

Nu is overigens helemaal geen gigagetal nodig in die PLL deler door RFREQ (de deler) om op 114 MHz uit te komen, want dat kristal met een getal tussen de 43 en 49 vermenigvuldigen doet je daar al in dat VCO-gebied landen. Jawel, maar de rest van dat gigagetal zijn allemaal cijfers achter de komma en die bepalen of er bij het delen pulsen worden weggelaten, zodat je ook gemiddeld tussen de hele vermenigvuldigingsgetallen kunt uitkomen. Vroeger in de TTL hard logic-tijd gebeurde dat al met zogenaamde "rate multipliers". Dat is een kreet die niet het verschijnsel omschrijft dat veel jonge vrouwen vertonen bij het vorderen van hun leeftijd, maar het weglaten van pulsen en aldus een frequentiedeling maken, weliswaar wel met jitter op de output, daar ontkom je dan niet makkelijk aan.

Vervolgens wordt de middels die speciale "rate multiplier" door de PLL gestabiliseerde VCO-frequentie tussen 4,85 en 5,67 GHz dan weer gedeeld door een getal dat het product is van een klein high speed prescaler-delergetal HS_DIV genaamd, dat verplicht naar keuze 4, 5, 6, 7, 9 of 11 moet zijn, en een gewone langzame domme deler verder genoemd N1, die kan delen door 1, dat is geen opvallende prestatie, en verder door alle even getallen tussen 1 en 129, en daarmee kom je dan uit op een frequentie, die de fabrikant opgeeft, met als laagste 10 MHz; en de amateur met naast-zich-neerleggen van reproduceerbaarheid en andere specificaties, op minder dan 3,5 MHz; en aan de

bovenkant 160 MHz voor de mij geleverde CMOS uitvoering C1, andere duurdere uitvoeringen gaan hoger in frequentie, doorlopend tot 945 MHz.

Een mond vol allemaal, dat wel. Maar ik wil niet hoger, want dat doet de ionosfeer ook niet. De eeuwige ruisvelden interesseren me pas na SK. Vast en zeker wel als mij dan 73 niet lesbische maagden worden toegewezen (Ja zendamateurs krijgen er drie extra). Overigens heb ik de controller geschikt gemaakt voor IC's tot 945 MHz. Er kan dus zonder modificaties een duurdere snellere Si570 worden ingesoldeerd.

Mu kun je die RFREQ deler van de VCO op rond 5 GHz niet maar raak sturen, dat kan slechts plus en min 3,5 kHz per MHz. Kom je daarbuiten dan moet je eventueel de RFREQ en de delers HD_DIV en N1 anders instellen en als dat laatste niet nodig is, in ieder geval de VFO stop zetten, en weer op gang brengen met de nieuwe frequentie als centraalfrequentie voor het 3500 ppm verstemgebied. Dat komt omdat de VCO bij grotere verstemming dan lock verliest door die weggehaalde pulsen waarmee de jitter dan te groot wordt om lock te kunnen handhaven.

Een heel gedoe dus, vooral als je een doorlopend afstembare VFO wilt maken, en hoe stel je al die parameters in?

Wel, dat gaat met de I2C bus. Dat zijn twee draden voor respectievelijk data en klok, SDA en SCL genaamd, waarmee je data kunt zenden en ontvangen, van en naar het IC, zodat je maar twee aansluitingen daarvoor hoeft te solderen op de uitgetrokken vliegenpoten soldeerplekjes van het IC. Dat lukte zelfs mij nog, wel pas na het snel drinken van 3 Bols jonge klare. Als je dat op het juiste moment doet gaan je handen trillen in tegenfase en met dezelfde amplitude van wat ze zonder borrel doen, en dat geeft dus rust en zekerheid.

Zelfbouw

Je hebt diverse categorieen.

1. Voor beginners bij voorkeur maximaal een jaar of 12 oud, een leeftijd waarop de gemiddelde amateur doorgaans blijft hangen, qua intellectuele ontwikkeling op zijn hobbygebied: Een kitje, uiteraard met printje, alles zit **erop** en **eran** en ze steken de onderdelen **erin**, in volgorde van hoogte, solderen die vast, maken kortsluitende onbedoelde soldeerbruggen, zetten de helft van de IC's 180 graden gedraaid, vloeken als een ketter - zoals de ouden zongen piepen de jongen - als een onderdeel, dat verder correct is, niet in de gaatjes van de print past; en als je geen pech hebt, zit minder dan de helft van de elco's verkeerd om. En de weerstanden zien er allemaal hetzelfde uit met zo'n bandje kleurcodes waarmee militairen doorgaans trots rondlopen, of ze veroordeeld zijn, a la Banning, of generaalmajoor buiten dienst zijn (soort troostprijs), maakt niet uit, als je kleurenblind bent.

Spanning erop, gelukkig heb je de "idiotdiode" in serie met de voeding toevallig goed gemonteerd, na korte tijd wat ontploffingen al of niet gepaard gaande met lanceringen en zo niet, dan heb je toch een probleem want het werkt niet, en dat nog wel terwijl je zo trots was dat je de hele zaak in 3 uur 45 minuten en 56 seconde in elkaar had zitten. Kampioen homebrewer zag je in je fantasie al ingelijst achter glas aan de muur hangen, naast je inmiddels vergeelde zwemdiploma.

Helaas komen weinigen verder dan het stadium van de 12 jarige.

2. Schemaatje nabouwen.

Dat is het volgende stadium. Je plakt volgens schema aanelkaar en je kijkt of het werkt. Mooi. Het oude zendamateurisme bestond uit voortrekkers die, al of niet met commerciele motieven, schema's publiceerden die je kon nabouwen. Als je dat een beetje handig deed, korte draden en zo, dan werkte dat, tenzij je een of ander omhooggevallen minkukel had die wat publiceerde om belangrijk te lijken, voor zijn eigen status dus, dan was je zuur. Sommige onderdelen vroeg je je van af of het ook wat anders mocht zijn, de verkoper in de radiowinkel gaf een raad die je overhaalde zijn winkeldochters af te nemen.

3. Plakwerk

Plakwerk is weer de volgende stap. je bouwt een VFO (dit artikel bijvoorbeeld) en ergens anders vind je weer een eindtrap, filters, een antennetuner, een antenne en noem maar op, en aldus bouw je de ideale zender, zoals Jan PAoHAM SK destijds verwoordde: "Het bittere einde". Die zond alleen CW als hij nog niet gegeten had en trilde door de honger, dat bevorderde op een wonderbaarlijke wijze zijn seinsnelheid op een straight key, daar kan ik van getuigen. Ik probeerde dat ook wel met een flinke scheut Jonge Klare, maar dat hielp niet, in tegendeel alles valt lam en ik kan zowaar dan SMD solderen.

4. Dan krijg je het echte home brew stadium, dat wil zeggen dat je je eigen genetisch be-invloede ei legt en thuis uitbroedt. De ideeen komen door een doos junk op de grond om te keren en ernaar te kijken. Als je weten wilt hoe dat proces gaat, kan dit artikel je daarbij helpen.

De I2C bus

Op die tweedraads I2C seriele bus kan elke deelnemer, optreden als master en als slave, en die rol aannemen. Heel gedoe als je dat in het algemeen voor veel deelnemers wilt regelen die elk op hun eigen houtje ineens master willen worden. Soort parlement in Belgie, stel ik me zo voor. Alleen spreken ze wel dezelfde I2C taal, dat is weer een voordeel. Na dit opruiende geleuter wat nuttige gegevens:

Een apparaat is master als het de gegevensstroom initieert, dat kan zowel een datastroom naar hem toe als van hem af zijn.

Wij hebben hier alleen te maken met een slave (Si570) en de master (een microcontroller).

Iedere als slave optredende deelnemer heeft een intern adres, hier bij de Si570 staat dat vermeld in het typenummer op het IC- huisje in de tweede regel, de 6 cijfers. Als je niet juist adresseert omdat je dat niet kunt lezen luistert hij (terecht) niet. Dat is vast een gewenning om 73 maagden op de bus te adresseren. Je moet er niet aan denken dat die zich gelijktijdig op je zouden storten door gebrek aan de juiste adresseringsmogelijkheid, als je belangstelling voor een van hen toont. Overigens kun je het adres ook vinden door alle adressen te proberen, tot je er een vindt waarop de slave een laag bit (acknowledge) teruggeeft.

Het hele zaakje is open collector of open drain (neus dichtknijpen in dit laatste geval), en in rust wordt dat met pull up weerstanden op SDA en SCL hoog gehouden.

400 kbit/s is wel zo'n beetje het maximum dat je op deze wijze kunt transporteren. Je zit immers met een niet te kleine pull up om de stroom beperkt te houden als je een 0 zendt, en de capaciteit van de bus als je de stroom stopt. Beetje primitief allemaal, maar ja, het is een Philips patent, wat wil je, dat in ieder geval wel printruimte beperkt houdt tot 2 aansluitingen voor de besturing en het aantal drievoudige jonge klares dus ook, zodat ik dit nog kan opschrijven.

Datatransport: data moet stabiel hoog of laag blijven zolang de kloklijn hoog is. Schrijven van data doe je dus tijdens klok laag en lezen van data tijdens klok hoog.

Er is een start- en een stopconditie voor datatransport, die het begin en einde van een datatransport aangeven. Startconditie is wanneer de klok hoog is de datalijn laag gaat. Stopconditie is wanneer de kloklijn hoog is de datalijn hoog gaat.

De master is master doordat hij een startconditie maakt (bus busy) en blijft dat tot hij een stopconditie afgeeft (bus idle).

Bij transport tussen master en slave bepaalt de master de klok, de slave kan echter de klok langer laaghouden, als die het allemaal wat te snel vindt gaan, en aldus het transport vertragen. Clock stretching heet dat.

In tegenstelling tot asynchrone communicatie wordt hier het MSBit van een byte het eerste verzonden.

Na 8 bits verzonden door de master geeft de slave op de negende klokpuls een (laag) acknowledge-bit af. Als dat niet zo is is het transport mislukt.

Leest de master de slave, dan moet de master na elk ontvangen byte een ack(nowledge) zenden (laag bit), behalve na het laatste ontvangen byte, teneinde de transmissie te verbreken.

Ontbreekt de acknowledge dan moet het transport verplicht worden beeindigd.

In ons geval is de controller altijd de master en de Si570 de slave

We beginnen met een 7 bits adres te zenden van de slave, gevolgd (LSB dus van een byte) door als achtste bit een 1 voor schrijven gewenst (master naar slave) en een 0 voor lezen gewenst van slave naar master.

Dat moet door een acknowledge worden gevolgd, door de ontvanger verzonden, waarna in geval van lezen de datastroom de andere kant (van slave naar master) op gaat.

Master moet elk ontvangen byte (behalve het laatste) van een ack voorzien. Stopcondtie beeindigt vervolgens het data transport. De master mag als bijzonderheid een stopconditie overslaan en met een start een andere of dezelfde slave aanspreken.

We beginnen met breien

Nu is het zaak zo snel mogelijk de theorie te testen, dan weet je namelijk of je die goed begrepen hebt, en bovendien kun je hem als het werkt dan zo snel mogelijk weer vergeten, om je hersen- werkgeheugen weer ergens anders voor te gaan gebruiken.

Dus ik ga in de junkbox graaien, volgens de xyl is het hele huis een grote junkbox, maar die heeft geen weet van het feit dat ik weet waar de trafo's liggen. Ik zie een aardig grijs bejaard exemplaar liggen, dat als opschrift meldt 8V 1,8 VA. Dat is qua spanning en stroom aan de krappe kant, maar, zoals bekend: "Het geluk is met de dommen", dus wat let me, succes verzekerd op grond van die wijsheid.

De voeding

Ik schat dat de hele schakeling wel op een half euroformaat (niet die munt, want die krimpt in waarde waar je bijstaat, maar) gaatjesbordje kan, dus ik begin maar links onder in de hoek om het netspanningsgebied op de print beperkt te houden. Kroon"steentje" erbij, eilandje tussen de netspanningpoten daarvan, met een aangesoldeerde draad eraf rukken om een wat langere kruipweg te creeren, netsnoer

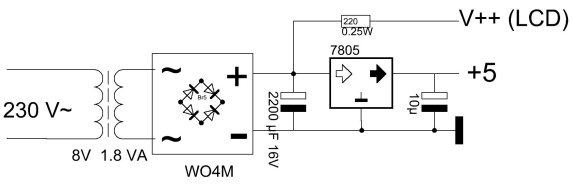


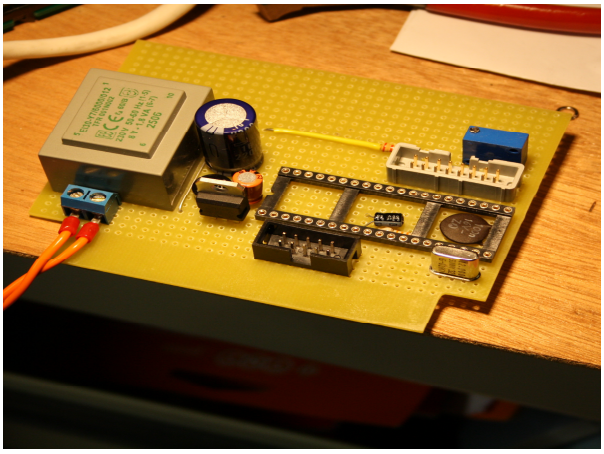
fig 2 Voeding

eraan. Even meten: Ja hij doet het. Hoe bestaat het. Vervolgens elco uit de grijpvoorraad 2000uF 16V, altijd genoeg maar minder heb ik niet, en een brugcel erbij plus een 7805 in TO220 huis zonder heat sink, die immers met dat prutsvermogen onnodig is. Afsluiten met 10uF 10V, allemaal grijpvoorraad, en kijken of dat schakelingetje, toegelicht in fig 2 werkt.

Ja het werkt. Goed zo, eerst een dot smeltlijm op de netaansluitingen aan de printzijde gedoteerd, want ik heb inmiddels een zwak hart door alle in de loop der jaren ontvangen opdonders van elektra en ik wil dit projectje wel afmaken voor het zeis vasthoudende geraamte, geheten Magere Hein, de benen binnenkort onder me uit maait. Hoe zwaar kun je het belasten voor de 5V geen 5V meer is volgens de scope, en 100 Hz deukjes begint te tonen? Dat probeer ik nog maar niet want er komt nog een LCD back lite (schaalverlichting, ze zullen lite light bedoelen) aan de ruwe plus in het schema aangegeven met te hangen. Dus "Op hoop van zegen".

De processor

Vervolgens een voet voor een controller gemonteerd. Zie fig 3 voor het schema. Voorlopig begin ik maar met de AT89S8253, uit de grijpvoorraad, niet de snelste in die voorraad, zeker niet, maar ik ben gewend aan de instructieset in assembler en de architectuur. Wellicht schakel ik later over naar wat anders door de software te porten, als dat nodig mocht blijken. De ontkoppeling, en de wellicht overbodige power-up reset worden tussen de high heeled voet (kunnen die 73 bloteplatvoetmaagden niet tegenop) en de print gemonteerd. De navelstreng ICP om te programmeren aangebracht, een kristal van 12 MHz voor een beetje peper in de anus van het werkpaard, en vooruit met de geit ingesteld op dubbele snelheid, op door de fabrikant voorgeschreven wijze. Ook maar gelijk nog een 16 pins boxed header met contrastregeling op port P2 gemonteerd van de controller voor de LCDisplay.



Eerst meten: staat er 5V tussen pen 20 en 31, repectievelijk 40 van de controllervoet; staat de ruwe plus (meter 10V knalt in de hoek) op pen 15 van de LCD-header en nergens anders op. Staat er 5 volt tussen pen 1 en 2 van de LCD voet. Dat gedoe. Ja, allemaal goed. Netspanning eraf, processor en LCD in hun voeten steken. Denk erom pen 1 van de processor is waar de nok in het huis zit en die moet zover mogelijk van het kristal op pen 18, 19 zitten. Niet een halve slag draaien dus, we zijn geen 12 jaar meer. Ook opletten dat je bij het plaatsen van de processor de pootjes niet ombuigt op de wijze waarop een kat door zijn voorknieen gaat als die op zijn buik gaat liggen. Spanning er weer op. LCD is verlicht, mooi, en verder gebeurt er niks. Weerstand van 47 ohm even tussen 5V en aarde zetten, en met de scope eroverheen kijken of de 100 mA extra belasting op de voeding geen deuken veroorzaken in het pakje boter van 5V, terwijl de LCD instaat. Ook even de spanning meten

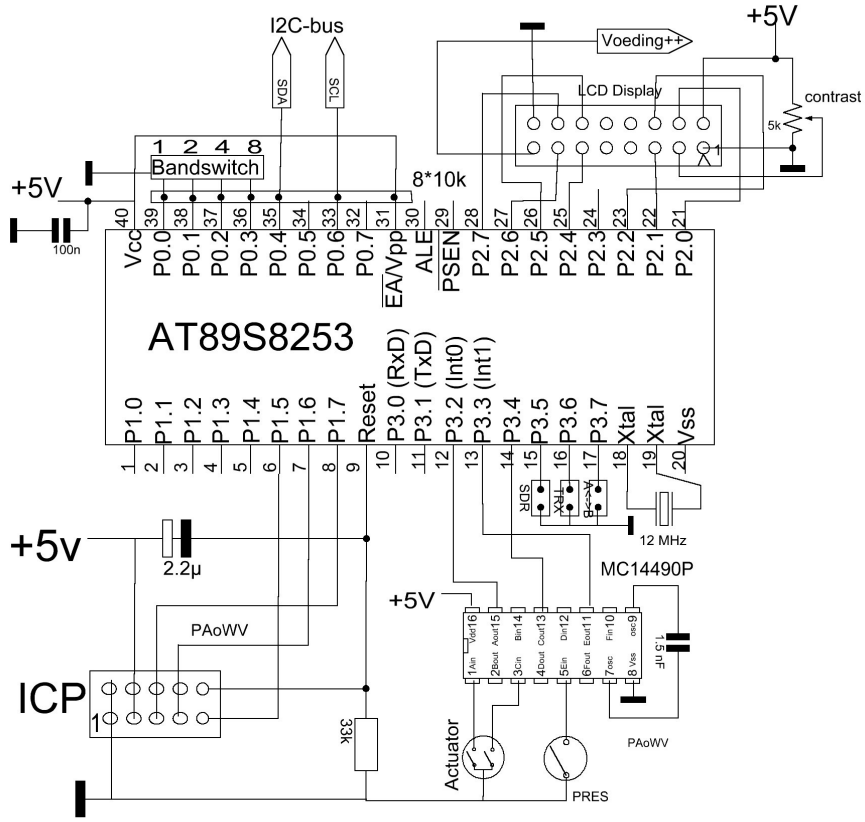
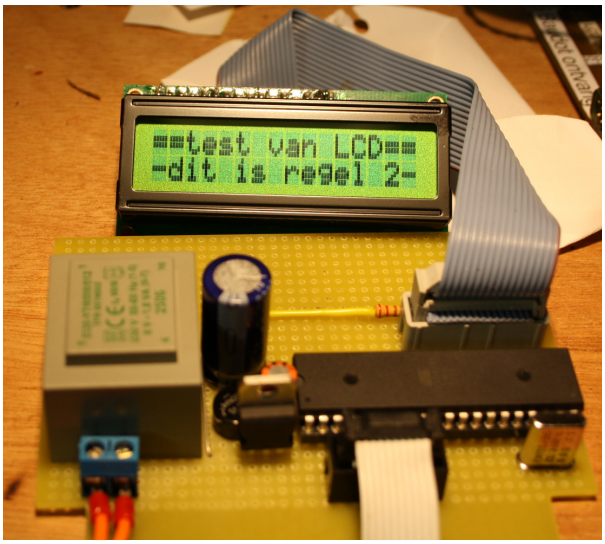


fig. 3 de processor

over de 220 ohm serieweerstand, dat geeft een indruk van de totaalbelasting van de voeding.

Ik zit u hier geen fabeltjes te vertellen, ik ben immers geen politicus, ter adstructie daarvan foto1. Het IC wordt via de ICP- connector vervolgens zo geprogrammeerd dat het niks anders doet dan P1.0 periodiek hoog en weer laag maken. Aan die poot is een frequentieteller gehangen, en daaruit blijkt bij mijn shacktemperatuur de kristalfrequentie van de 12 MHz kristallen die je voor 40 ct kunt kopen in Bentheim, kort na inschakelen ruim 6 kHz hoger dan nominaal aangegeven. Voorts heb ik een testbericht geprogrammeerd voor de LCD, die na instelling van de contrastpotmeter ook blijkt te werken.

LCD in een vroeg stadium aansluiten is nuttig als je wilt debuggen omdat iets niet werkt zoals je dacht dat het zou moeten werken. Je kunt er posities in RAM mee bekijken.



Stukje programma geschreven dat 9 RAM-posities die elk met een BCD 0 - 9 gevuld worden netjes op de display zet dat wil zeggen met onderdrukking van leading zeroes en met een punt om de drie cijfers als die wel worden afgedrukt, gevolgd door de eenheid Hz. Daar is de eerste regel net mee vol. Dat werkt ook.

De actuator

Nu is het de bedoeling dat de frequentie met een knop gewijzigd kan worden. Dat kan met een actuator, dat is een schakelaar die bij draaien van de knop steeds aan/uit gaat, en daarbij ingebouwd een tweede schakelaar die dat 90 graden verschoven in de positie doet, waardoor je kunt weten of de knop rechts- of linksom draait. Die schakelaars denderen (bouncen) en dat is lastig, maar Bournes de fabrikant van dit type ECW1J-B24-AC0024L, zet in de specificatie dat je een debouncing IC kunt toepassen van Motorola de MC14490P. Je kunt ook zelf een optische actuator maken door een schijf met zwarte en doorzichtige streepjes op een cirkel door een IC dat je in een oude optische muis kunt vinden, te draaien. Heb ik even nog niet gedaan, dus voorlopig hangt er een actuator aan met een debouncer IC. Een derde mogelijkheid is bij Farnell een daar op voorraad liggende optische encoder te kopen, type ENA1J-B28-L00100L, die heeft 100 stappen per omwenteling en je hebt geen debouncing IC nodig want die bouncen niet, maar ze hebben wel 25 mA 5V voeding nodig. Die heb ik aangeschaft en die werkt ook, de MC14490 is dan niet nodig, maar die heb ik laten zitten, daar kun je namelijk ook een benodigd drukknopje mee debouncen, hoewel dat met software ook gekund zou hebben, zoals met de verderop je bespreken bandschakelaar is gebeurd.

Een poot van de debounced actuator geeft een externe interrupt op P3.2, als die optreedt wordt in de interruptafhandeling gekeken op P3.4 om vast te stellen of die laag of hoog is wat de draairichting bepaalt. Is die rechtsom, dan wordt de BCD 9 cijferige frequentie - die op de display vertoond wordt - verhoogd, anders verlaagd. Dat stukje daarvoor geschreven software is dus snel te testen. Niet alles gaat in een keer goed, maar aan de hand van wat op de display te zien is kan beredeneerd worden wat er niet goed zit en dat is vervolgens dan te corrigeren.

Nu is het zo dat de VFO niet op 1 Hz stabiel zal zijn, (scheelt overigens weinig) maar toch is de uitlezing tot op 1 Hz resolutie gemaakt, opdat je bij verdraaien van de afstemknop de frequentie ongeveer met dat bedrag uiteindelijk minimaal kunt verhogen of verlagen. Als de rate multiplier 28 bits achter de komma heeft, geeft dat een minimaal verschil van 2^-28 *114 MHz op de VCO gemeten, dus 0,425 Hz in VCO frequentie en dat is dus dan ongeveer 1E-4 Hz per MHz.

Nu is een VFO afstemmen in stapjes van 1 Hz leuk als je iets verstemt maar niet als je een flinke ruk wilt verstemmen. Bij 24 kHz kun je al een boormachine op de afstemknop zetten om de nieuwe QRG te bereiken. Dus 24 Hz per knopomwenteling van de mechanische versie is 1000 omwentelingen, Bij de genoemde optical encoder 240 omwentelingen. Daarom zijn verschillende stapgrootten gewenst. De stapgrootte zit in een byte genaamd delta in de controller geprogrammeerd, en die bevat ergens een en slechts een bit 1, de rest van de 8 bits dus 0; en de positie van die 1 geeft de stapgrootte aan; helemaal rechts 1 Hz, helemaal links 10 MHz. Bij elke positie naar links vergroot de stap een factor 10. Bij testen heb ik daar voorlopig maximaal 1 MHz van gemaakt, want bij 100 stappen per omwenteling kun je dan in een omwenteling al 100 MHz verstemmen als je snel draait. Draai je langzaam aan de afstemknop dan moet hij per Hz verlopen en naarmate je sneller aan de knop draait moet de stapgrootte groter worden.

Dat kan door een timer_0 interrupt te gebruiken die een 8192 teller_0 verhoogt bij elke teller-overflow die een interrupt genereert (244 per seconde) en dan de tijdteller van een byte in RAM verhoogt. Van de actuator wordt in zijn externe interruptafhandeling de hoogte van die tijdteller bekeken en teruggezet op 0. Is hij niet hoog dan wordt er kennelijk snel gedraaid en wordt de 1 in delta naar links gezet. Zo ook bij hoge waarden van de tijdteller, dus

langere tijd tussen twee actuator clicks, wordt de 1 op een navenant lage positie gezet. Als de teller op 255 komt. dan loopt hij niet verder meer maar blijft daar staan tot de actuatorinterrupt dat constateert en hem op 0 terugzet.

Conversie naar binair Nu moet er met die frequentie die in BCD in het geheugen en op de display staat worden gerekend, daarom is een binaire representatie gewenst. Na elke schaalwijziging op de LCD wordt die berekening uitgevoerd. Dat kan gebeuren door de getallen 1, 10, 100 ... 10^8 binair in het programma geheugen te zetten, die getallen te vermenigvuldigen met de bijbehorende BCD waarde tussen 2 en 9 en dat hele zaakje op te tellen. Nu is er in de controllerchip programmageheugen zeer ruim voorradig, terwijl de snelheid niet je _dat is, daarom is dat vermenigvuldigen hier te voorkomen door binair de waarden van de bovengenoemde vermenigvuldigingen in een tabel op te nemen zodat slechts 4 bytes brede optellingen van posities uit de tabel als taak resteren. Die tabel is in assemblerlisting aangemaakt door een daartoe geschreven programma in C genaamd multitab.c en op mijn website te vinden onder de zelfbouwlink Si570_VFO. Dat vermijdt rekenfouten, typefouten en vergissingen, wat voor_komt dat soms de VFO niet op de frequentie zou staan die de display aangeeft. Zeker bij een zender dient dat voorkomen te worden.

De maximumfrequentie van de Si570 in CMOS uitvoering is 160 MHz, en dat is te noteren binair in 28 bits, een byte is 8 bits dus we noteren de waarden in 4 bytes. Dat zou zelfs voldoende voor ruim 4 GHz zijn, zodat de 945 MHz limiet van de andere type Si570 geen probleem vormen.

We zijn nu dus zo ver dat als we de frequentie als aangegeven op de LCD omhoog en omlaag draaien, die bij elke wijziging naar wens kan worden omgezet in een 4 byte binair getal.

De interfacing naar de Si570

De Si570 draait op 3,3 V, dus een extra low drop stabilisator LM1117 3.3V voor de voeding is vereist. Zie fig 4.

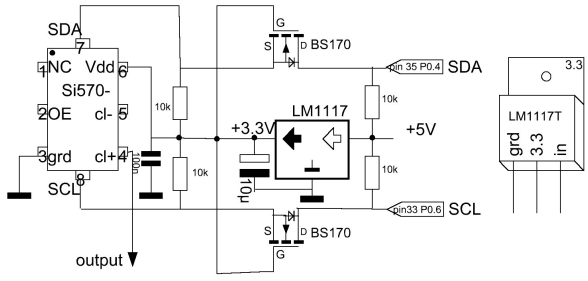
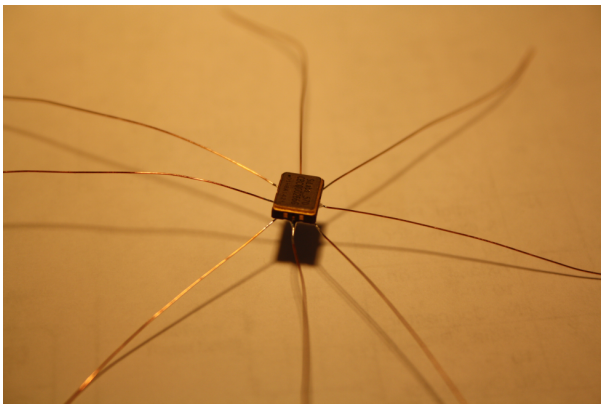


fig 4 Si570 voeding en I2C interface

Niet alleen de voeding, maar ook de bidirectionele inderface van de I2C bus naar de processor moet in worden voorzien met een bidirectionele levelshifter 3,3 <-> 5 V. Philips geeft in een Application note AN97055 aan hoe dat kan met twee stuks BS170 N channel MOSFETs. Die zijn te koop bij Conrad onder bestelnummer 158950 en kosten daar 51 ct, als je er 10 koopt, wat ik deed. Je kunt ze namelijk ook, een of enkele parallel geschakeld, als QRP eindtrap tot 5 watt gebruiken.

Op de aansluitvlakjes van de Si570 worden adertjes gesoldeerd van gestript soepel netsnoer. Die zijn blank



koper en ongeveer 0,18 mm dik, foto2 toont het resultaat van hoe je van een vlieg een spin maakt. De BS170 moet je mee uitkijken wat aansluitingen betreft, die kunnen per fabrikant verschillen. Conrad leverde mij Fairchild en die is als je naar de draden kijkt in de richting van je ogen gehouden, en de platte kant van het MOSFET huisje naar beneden van links naar rechts Source-Gate-Drain. De drain komt aan de controllerkant.

Kortom, er kan van alles fout gaan dus we monteren eerst de 3,3 V voeding en de levelshifters, als 10 k pull up weerstanden heb ik staafjes gebruikt met acht 10 k weerstanden per stuk. De opstelling van die onderdelen op gaatjesbord is apart getekend in fig 5. Alle poten van port P0 zijn via zo'n pull up aan gehangen, nodig om die andere poten van P0 ook te kunnen gebruiken voor andere doeleinden.

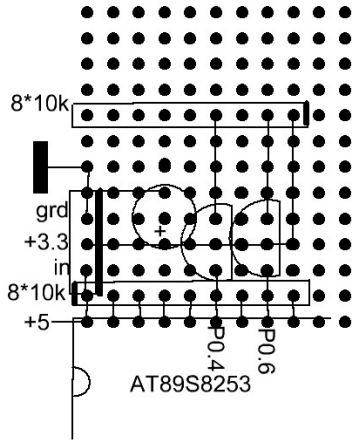


fig 5 Onderdelenzijde montage Si570 interface

Meten of de spanningen kloppen, vervolgens is er nog een testroutine gemaakt, die een poot van de controller op en neer haalt en kijkt of de andere poot van de controller geschakeld als input volgt. Ook die input en output verwisseld. Dat uiteraard als de SCL en de SDA aan de Si570 zijde van de level shifters zijn doorverbonden voor de proef. De routine zit in het diagnostic gedeelte van de software source listing onder de naam test_I2C. Na die controles wordt de Si570 pas aangesloten. In feite zijn er maar 5 draadjes die moeten worden aangesloten SDA SCL grd, Vdd en output. OE hoeft niet aangesloten, die is inwendig al geactiveerd, en een pen is not connected (NC), Er is bij dit model geen balansoutput Cl-, zodat er 5 aansluitingen overblijven. De 3.3 V voeding wordt dicht bij de Si570 ontkoppeld met een 100 nF keramisch C'tje. Teveel aangesoldeerde draadjes kunnen dus verwijderd.

Spanning erop, en frequentieteller eraan. Er komt bijna 15 MHz uit, dus er zit leven in de brouwerij. Tevens geverifieerd dat er op Cl- inderdaad geen output is. De temperatuurdrift na inschakelen is 200 Hz naar beneden voor hij stabiel wordt op nagenoeg 1 Hz.

Het I2C adres

Volgende stap is vaststellen van het adres van de Si570, dat zou uit de tweede regel van het typenummer moeten blijken, die luidt CBC000266G maar dat verhaal is me niet geheel duidelijk. Er wordt dus voor de voortgang toch benodigde assembler geschreven om de I2C bus als master aan te sturen. Vervolgens worden alle mogelijke 128 adressen afgeraffeld en gekeken of er antwoord komt van de Si570 in de vorm van een ACK bit. Dat adres wordt op de display gezet, dan weten we dat. Dat lukt en het blijkt hex 55 te zijn. Een getal dat ik niet in het typenummer herken.

Lezen en schrijven in de Si570 Vervolgens worden routines geschreven om een volledig commando te lezen en te schrijven, zoals de datasheet van de Si570 op blz 16 daarvan opgeeft. Daarmee worden bij wijze van test de registers 7 t/m 12 van de Si570 hun waarden uitgelezen en in hex op de display gezet. Ook dat werkt na enig debuggen. Dat levert in mijn geval A8C2A85D74F3. Volgens de specsheet is dat te herleiden tot snelle deler HS_DIV= 9, tweede deler N1= 36 . Het product van die twee is 324, zodat de volle range van de VCO een uitgangsfrequentieband van de Si570 levert tussen 14,969 en 17,5 MHz met die delerwaarden. De gemeten uitgangsfrequentie ligt in dat gebied, dus dat kan. De RFREQ, dat is het deelgetal van de VCO dat de Xtal frequentie levert, blijkt tevens uit het LCD scherm, Daarvan zijn 28 bits achter de komma dus het getal is hex 2A,85D74F3. Mijn teller wijst uiteindelijk 14,999880 MHz aan en drift een hertz op en neer op lange termijn. Daaruit is de kristalfrequentie van de Si570 te berekenen met de nauwkeurigheid van mijn frequentieteller; namelijk 324 maal de telleruitlezing levert de VCO frequentie en die delen door RFREQ levert de kristalfrequentie op. Dat wordt dan 114,29066 MHz wat ter controle van dit alles weer kan kloppen.

Omdat het kristal moet uitkomen in het VCO gebied is dat RFREQ getal dus minimaal 42 en maximaal 49 voor de komma, oftewel hex het gebied 0x2A tot 0x31. We hebben met 2 delers te maken HS_DIV en N1 die in cascade achter de VCO staan, die voor de uitgangsfrequentie zorgen, De deeltallen moeten gekozen worden om de juiste gewenste frequentie uit de VFO te krijgen met de VCO tussen de genoemde GHz grenzen. Daartoe heb ik een programma geschreven in C (vfomult.c), de output staat eveneens onder de genoemde link op de zelfbouwpagina van mijn website met de bestandsnaam delers.txt, dat alle mogelijke combinaties bevat van de snelle deler en de langzame erachter, met een minimumwaarde van het product gelijk aan 6. Dat loopt dan tot 945 MHz. Je kunt theoretisch ook 4 en 5 gebruiken als product, maar dan krijg je gaten in de dekking van de hoogste banden.

Het blijkt dan, voor het gebruikte CMOS IC, dat je niet alle kleine deeltallen kunt gebruiken want dan kom je met een product van HS_DIV (de snelle deler) en N1 (de langzame deler) beneden de 32, boven de 160 MHz uit, wat voor deze CMOS versie de bovengrens is. De ene deler kan op 6 toegelaten waarden staan, de andere op 65 verschillende waarden, zodat we totaal 390 combinaties hebben. Die combinaties zijn gesorteerd op grootte van het product, dat tot 945 MHz tussen 6 en 1408 uitkomt; en dat geeft dan de frequentiegebieden aan die bij die combinaties horen. Dat zijn er minder dan 390, want er zijn een aantal combinaties waarvan het product hetzelfde is. Die dubbele instellingen waarvan het product hetzelfde is worden in het programma vfomult.c verwijderd, met dien verstande dat het product met de hoogste waarde van de factor HS_DIV blijft staan, zoals aanbevolen in de Si570 data sheet, waarna er 276 combinaties overblijven die in de lijst delers.txt staan. De lijst zou nog veel verder kunnen worden opgeschoond, als we zien dat bij alle entries de frequentiebereiken door hun boven en onderliggende buur-entries min of meer worden overlapt. Nadere inspectie van de tabel delers.txt leert dat slechts 34 paren deeltallen volstaan om het gehele gebied van 3,5 tot 945 MHz te beslaan met minimale overlap. Dat is niet verbazingwekkend want de VCO verhouding maxfreq/minfreq = 5, 67/4,85=1,17 en de verhouding 945 MHz / (1 ,17 *3445kHz)=235 zodat het theoretisch minimum aantal noodzakelijk paren n volgt uit 1,17^n=235 met n dus log (235)/log(1,17)= 35.

De VFO outputfrequenties zijn dus met overlap bereikbaar. De specsheet raadt in geval van overlap aan om de laagste VCO frequentie te gebruiken (het kleinst mogelijke deeltal N1*HSDIV dus) omdat de dissipatie (en dus ook de drift) dan wat geringer is. De specsheet zegt voorts dat de minimaal bruikbare frequentie 10 MHz is, waarom ze dat beweren weet ik niet want de frequentie wordt bepaald door de twee delers op de maximaal mogelijke waarde te zetten. Ik zie geen plausibele reden dat dat niet zou mogen; kunnen doet het zeker. De minimale frequentie die de Si570 afgeeft is dus 3445 kHz. Voor SDR apparatuur die graag 4 maal de nominale frequentie aangeboden krijgt, kun je dus tot een kwart van dat bedrag nemen, ten gevolge van de externe Johnson 4-deler. De radicale Imamtoespraken op 1007 kHz

van de Nederlandse publieke omroep in de Arabische taal hoef je dus niet te missen als je deze VFO gebruikt voor SDR ontvangst.

Testen van de I2C sturing

Nu dit alles bekend is kan de I2C schrijfroutine worden getest door andere waarden in de 6 bytes reg7 t/m reg12 van de Si570 te zetten en te kijken of de frequentie uit de VFO dan klopt. Tevens kunnen die registers dan weer worden uitgelezen naar de display als dubbel check met de I2C read routine.

De waarden kunnen geschreven worden, omdat bij het reeds eerder geteste teruglezen blijkt dat ze erin staan. Echter de frequentie aan de uitgang wijzigt bij een forse wijziging niet, omdat dat extra control inputs vergt die nog geprogrammeerd moeten worden in Si570 registers reg135 en reg137. Dat kost dan maximaal 10 ms onderbreking van het uitgangssignaal volgens de datasheet. Dat geldt voor elke wijziging van RFREQ die groter is dan 3500 ppm, Een grotere wijziging vereist opnieuw instellen van de centraalwaarde van RFREQ en eventueel de delers, middels opdracht via register 135 en 137 van de Si570.

Je kunt de VCO en dus RFREQ ook binnen een tolerantie van 3500 Hz per MHz zowel naar boven als naar beneden wijzigen, zonder enige andere wijziging dan opnieuw schrijven van (een deel van) RFREQ. RFREQ ligt tussen 0x2A en 0x32 dat bepaalt stappen van 114 MHz in de VCO frequentie. De 28 bits achter de komma bepalen daar een deel van.

We werken bij de berekeningen binair, de amplitude van de toegelaten deviatie in RFREQ voor 3500 ppm blijkt op minimaal 3500E-6*RFREQ_min = 0x0.2605B91 te kunnen worden berekend. Dit betekent dat we RFREQ zonder onderbreking van de output, 'on the fly' kunnen wijzigen mits we binnen deze deviatie van de laatst ingestelde nominale waarde blijven. Zou je te ver verstemmen dan verliest de VCO lock. Op een teller op de uitgang van de Si570 is dat duidelijk waar te nemen, omdat dan ineens de rockstabiliteit van het outputsignaal aanzienlijk vermindert.

Grotere sprongen in frequentie dan 3500 ppm moeten gebeuren door de VCO stop te zetten door schrijven van register 137, de nieuwe registers van delers en RFREQ te laden, voorzover die gewijzigd zijn, de VFO weer aan de gang middels register 137 en daarna de newfreq bit in register 135 te activeren. Blijf je binnen de band met dezelfde delerwaarden, dan volstaat het dus, omdat de delers en RFREQ op het randje van het toelaatbare gebied correct is ingesteld, om de VFO stil te zetten, te starten en een setfreq procedure te doorlopen, aangezien er geen registers te wijzigen zijn.

Dissipatie en drift

De specsheet raadt aan om in geval van keuzemogelijkheid HS_DIV zo groot mogelijk te kiezen en N1 dus zo klein mogelijk om een gewenste productwaarde HD_DIV*N1 te maken. Dat zou minder dissiperen, en derhalve de temperatuurvariaties in de chip verminderen wat de stabiliteit van de output weer ten goede komt. Voorts wordt geadviseerd om bij keuzemogelijkheid (die strijdig kan zijn met de voorgaande eis) de VCO in het lage deel van het frequentiebereik te houden, dus het laagste product HS_DIV * N1 kiezen dat mogelijk is om de gewenste frequentie te verkrijgen, als er meerdere keuzes mogelijk zijn.

Om een indruk te krijgen meet ik de verbruiksstroom bij dezelfde lage VCO waarde (= lage RFREQ=0x2B,0) voor delers 4 maal 22 en voor 11 maal 8, beide met het product 88 dus, kort achter elkaar. Niet nodig die meting, maar we hebben onze zendmachtiging voor het doen van onderzoek en zelfontplooiing, niet voor radiowedstrijdjes en slap geOHzwam in de microfoon van een jappenkoopbak met default menuinstellingen en een antenne uit blisterverpakking eraangeknoopt.

Vier mogelijkheden zijn er: VFO boven en beneden in zijn bereik, en de delers als genoemd, dat geeft resultaten die in een tabelletje zijn opgenomen. Het blijkt dat de uit de meetresultaten berekende kristalfrequentie door de temperatuurvariaties op de chip ten gevolge van de variatie in dissipatie wel 40 Hz op en neer fietst. De twee delers samen door 88 laten delen met de VCO op dezelfde plek geeft al een verschil van 20 Hz, door de andere delerdissipatie terwijl de freqentie dan gelijk zou moeten blijven.

Bij hetzelfde deeltal, de VFO van links naar rechts in zijn bereik verplaatsen geeft ook dissipatieverschil, en de uit de uitgangsfrequentie berekende kristalfrequentie verschilt dus, waaruit het kristalverloop blijkt, dat in de tabel staat opgegeven. De tabel is gegenereerd, door de controller zo te programmeren dat hij cyclisch de frequenties kiest voor ruim 16 seconde elk. Onmiddellijk na een wijziging is de drift het grootste, de opgegeven waarden zijn eindwaarden vlak voor de volgende wijziging.

HS-DIV N1	output MHz	mA	Xtal MHz
11 * 8	63,639130	94	114,29068
4 * 22	63,639115	90	114,29066
11 * 8	55,846581	80	114,29068
4 * 22	55,846564	75	114,29064

We zien hieraan dat er verloop is ten gevolge van temperatuurwijziging op de chip bij dissipatie wijzigende instellingen. Het kristal verloopt 40 Hz tussen de twee uitersten. Dat heeft dus niks te maken met de omgevingstemperatuur. Het zou dus kunnen _voorkomen dat bij toename van de VFO frequentieinstelling, bij wijziging van een instellingsgrens de frequentie daalt in plaats van stijgt. Om dat effect

te minimaliseren lijkt het geboden inderdaad zich te houden aan de aanbevelingen van de fabrikant, zoals genoemd, teweten: Houdt de VCO frequentie laag en HS_DIV hoog in geval er keuzes mogelijk zijn tussen verschillende instellingen voor een gewenste outputfrequentie.

De conclusie is dat er dan meer dan het noodzakelijk aantal van 35 delerinstellingen moeten zijn om de VFO aan de lage kant te houden, zonder grote sprongen van hoog naar laag in het bereik. Van de andere kant zijn de 10 ms signaalloze gaatjes bij wijzigen van de frequentie >3500 ppm een noodzakelijk kwaad, dat we liever niet te vaak hebben, maar het VFO bereik opdelen in meer bereiken, zal het aantal gaatjes dat elke 7000 ppm optreedt over het gehele afstembereik niet beinvloeden. Dat is een constante.

Ik acht stabiliteit belangrijk, mede door de geplande toepassingen, zodat de werkwijze is geweest zoals de volgende paragraaf Bereikenkeuze omschrijft.

Bereikenkeuze

We kunnen alle mogelijke HSDIV*N1 producten sorteren op grootte Dat bepaalt het frequentiebereik dat bij elk product hoort, dat is immers (4.85 tot 5,67 GHz)/(HSDIV*N1). Een aantal producten HSDIV*N1 zijn onderling gelijk terwijl de factoren N1 en HSDIV verschillen, zoals we zojuist zagen in het meetvoorbeeld. Als de producten gelijk zijn wordt alleen die met de hoogste HSDIV genoteerd, overeenkomstig advies fabrikant. Dat levert dan gesorteerd 276 producten op, met overlappende bijbehorende frequentiebanden.

Vervolgens willen we niet tijdrovend gaan zoeken door beurtelings te vergelijken, welk laagst mogelijke product HSDIV*N1 bij onze gewenste binair in 4 bytes genoteerde frequentie past, maar willen we dat in verband met het minimaliseren van zoektijd geindexeerd doen.

Lage frequentiebanden (dus met grote HSDIV*N1) liggen dicht bijelkaar. Alle banden hebben immers de range dat de hoogste frequentie 5,67/4,85 (de VCO grenzen) maal de laagste is. De deviate van de nominale frequentie van 3500 ppm willen we aan beide zijden van de berekende range aftrekken, zodat een nominaal gekozen frequentie altijd gedevieerd kan worden binnen de 3500 ppm grens zonder een door HSDIV*N1 gekozen set bandgrenzen te overschrijden. Dat spaart ook een controle daarop en dus rekentijd uit. De nominaal instelbare VCO grenzen worden dan dus

4850*(16)= 4867 MHz en 5670*(1-3500E-6)= 5650 MHz

Bij elk van de 267 tussen 6 en 1408 op grootte gesorteerde producten van HSDIV is nu de bijbehorende band bekend. Die namelijk 4867/(HSDIV*N1) tot 5650/(HSDIV*N1) MHz. Vervolgens moeten we een van die producten kiezen bij een gekozen VFO frequentieinstelling in zo weinig mogelijk tijd. Dat gebeurt daarom dus geindexeerd. Acht bits uit de eerste eerste twee bytes van de 4 byte binair genoteerde VFO frequentie worden daarvoor genomen. De grootste range van de index wordt gevonden als we 8 bits nemen na de eerste 6 bits, mits die eerste zes alle 0 zijn. Dat wordt dus een range van lagere frequenties. Nagezocht middels het programma rfreq.c is dat in dat geval aan de volgende voorwaarden altijd is voldaan:

a) Elk met een product N1*HSDIV gekozen frequentiegebied valt geheel binnen een exemplaar van de index, Je hebt dus nooit dat een door een index aangegeven frequentiebereik niet een geheel door een index aangegeven subbereik bevat. Een indexbereik is 000000[8bitsindex]en dan aangevuld met 18 bits tot het totaal van 32 bits met bits in de range allemaal 0 tot allemaal 1. Dus in dit voorbeeld: 000000[8bitsindex]000000000000000000 tot 000000[8bitsindex]111111111111111111
b) Er hoeft niet gecontroleerd te worden bij verstemming binnen 3500 ppm of de VFO grenzen worden overschreden
c) Voor elke frequentie is altijd de hoogst mogelijke HSDIV in combinatie met de laagst mogelijke N1*HSDIV voorzien.
d) Elke gekozen frequentie wordt gedekt door deze werkwijze.

Met 256 op deze wijze gekozen indexgetallen bedek je echter niet de gehele frequentie-range tot 945 MHz, namelijk niet de producten van N1*HSDIV die kleiner zijn dan 80. Dat is boven 67,5 MHz In dat geval zijn de eerste 6 bits van de binair genoteerde frequentie ook niet meer alle 0.

Daarvoor moet dus een tweede index worden gekozen, voor het geval de eerste 6 bits niet 0 zijn. Die index bestaat uit het derde t/m het tiende bit van de frequentie. Er is dan dekking tot 945 MHz maar met een paar complicaties, namelijk dat de indexen 16, 48, 96 en 192 niet een frequentiegebied aangeven dat door _een product N1*HSDIV wordt gedekt. Daarom worden die indexen met een exemption-handler gesplitst in 2 bereiken en het daarbij behorende product genomen. Dat eist dus extra maatregelen en dus een iets bewerkelijker proces voor de frequenties boven 65 MHz.

Op deze wijze kunnen we zonder gok- en zoekwerk, direct geindexeerd de optimale combinatie van N1 en HSDIV vinden.

Kost wel tabelgeheugen maar daar zijn we ruim van voorzien in de controller, en het zoeken van de optimale HSDIV*N1 (zo laag mogelijk) wordt er zo kort mogelijk door.

Rekenwerk voor de controller

Elke entry in de tabel van producten N1*HSDIV bevat voor dat bereik de twee deeltallen, HS_DIV en N1 reeds gecodeerd volgens het vereiste format voor de Si570 registers reg8 en reg7. Naburige bereiken kunnen vaak hetzelfde deeltalpaar hebben en voorts zou elke entry een 4 byte getal moeten bevatten zijnde het product van de twee delers gedeeld door de kristalfrequentie. Dit laatste getal moet dan bij elke schaalwijziging op de LCD schaal vermenig-

vuldigd met de binaire representatie van de LCD schaalwaarde, die we eerder berekend hadden met behulp van de opzoektabel multitab en die bij elke schaalwijziging direct wordt berekend.

Nu is de kristalfrequentie van elk exemplaar Si570 wat verschillend zodat die niet in een vaste tabel kan worden ingeprogrammeerd in een microcontroller die met een willekeurig aangeschafte Si570 moet samenwerken. De vereiste RFREQ voor de schaalwaarde wordt namelijk berekend uit de formule zoals uit de tekening in fig 1 onmiddellijk blijkt

RFREQ= LCDfreq * {N1 * HS_DIV/Xtal}

Het totale deeltal N1 * HS_DIV dat tussen 6 en 1408 ligt voor alle versies van de Si570, is gecodeerd in de tabel. De RFREQ moet dan uit elke binair gecodeerde schaalwaarde op de LCD worden vermenigvuldigd met het totale deeltal N1*HS_DIV en gedeeld door de bij calibratie vastgestelde Xtal frequentie. Alles 4 of 5 byte breed.

De gang van zaken is uiteindelijk als volgt:

1. Bepaal de 32 bits [0,31] binaire waarde van de LCD schaalfrequentie. Als de schaalfrequentie een bandlimiet overschrijdt (bandkeuze wordt nog verderop behandeld) dan wordt de schaalwaarde teruggezet op de bandlimiet, alvorens de BCD frequentie binair te coderen.
2. Bepaal uit de binaire gerepresenteerde frequentie een index die 8 bits is door er de besproken hap uit te nemen.
3. Dit is de index in een tabel minder dan 255 lang, en die tabel geeft het adres van voor elk indexgetal een entry in een andere tabel prodtab van producten. Die laatstgenoemde tabel bevat per entry 4 bytes, dat zijn 2 bytes voor N1*HS_DIV en twee bytes die N1 en HS_DIV bevatten met de door de fabrikant van de Si570 voorgeschreven codering.
4. De binaire frequentie wordt eerst vermenigvuldigd met N1.HS_DIV uit de tabelentry, dat levert een waarde die niet veel warieert, omdat bij toenemende frequentie HS_DIV*N1 daalt.
5. De waarde van 1/Xtal voor de gebruikte Si570 is bekend (uit een te bespreken calibratiestap). Die waarde wordt vermenigvuldigd met het resultaat van stap 4.
6. N1 en HS_DIV worden er volgens de Si570 codering uit de prodtab tabel beschikbaar bijgeplakt.
7. Dit hele zaakje (6 bytes) gaat via I2C de Si570 in om de frequentie te programmeren.
8. Het verschil in frequentie met de laatste centraalfrequentie wordt bijgehouden, Indien de range van 3500 ppm wordt overschreven wordt de centraalfrequentie opnieuw op de nieuwe waarde ingesteld.

Multiplier

In ieder geval hebben we een multiplier nodig van 4 bytes breed. Twee getallen van 4 bytes vermenigvuldigen geeft een resultaat van 8 bytes. Dat proces kan ingekort als bytes 0 zijn. En zoals we op de basisschool leerden hoe je grote getallen moet vermenigvuldigen, zo werkt dit hier ook. We deden dat met de tafels tot 10. Hier echter is een 8 bits brede multiply instructie in de controller zodat we grotere happen in een keer kunnen vermenigvuldigen met de tafels tot 256. Voor twee 32 bits getallen zijn dan maximaal 16 multiplies nodig en de bijbehorende optellingen. Een en ander is geprogrammeerd, en om dat compact te houden wordt indirect geadresseerd, daarvoor zijn echter 4 adrespointers nodig, terwijl de gebruikte controller er maar 2 heeft. Maar die controller heeft 4 banken met pointers zodat een bank wisselen een tweede set van 2 pointers levert.

Testen en debuggen, tot een paar vermenigvuldigingen goed werken blijkt weer noodzakelijk.

Verdere stappen in de ontwikkeling

Een deelroutine is altijd nodig, namelijk om de zaak te calibreren waarbij 1/Xtal moet worden berekend. Dat delen van 4 bytes deeltal en 4 bytes deler gaat ook weer net zoals op de lagere school geleerd, maar hier is dat bitsgewijs geimplementeerd. Dat de kindertjes op de lagere school niet wisten dat je dan niet tien tafels van 10 stuks elk, maar slechts 2 tafels van 2 stuks elk uit je hoofd hoeft te leren, te weten de tafel van 0 en de tafel van 1, als je binair rekent, is maar goed ook, anders hadden we al demonstraties van 6 jarigen betreffende de kwaliteit van het onderwijs. Het tientallig stelsel schiet wel lekker op, maar wat dacht je van het 256 tallig stelsel waar ook een deelinstructie voor in de controller zit. Toch maar voorlopig niet gaan gebruiken, want ik wil een bruikbare VFO maken en geen rekenkundige worden. Bovendien is de calibratiestap eenmalig en die hoeft helemaal niet supersnel te zijn. Het quotient is 5 bytes en dat zijn ruim voldoende significante cijfers, om de factor 1/Xtal te berekenen.

Verder testen en breien

De vraag is hoe snel de hele wijziging gaat van twee frequenties die niet binnen de zelfde band en dus ook niet in dezelfde 3500 ppm range vallen. Dat kan onderzocht door het changed bit waarop de main routine normaal staat te wachten, en dat aangeeft dat aan de actuator is gedraaid, weg te laten en te vervangen door een spanningsomkering op een uitgangspen, waarvan de frequentie dan gemeten kan worden. Daaruit blijkt dat de hele rataplan inclusief de display van de frequentie en de diagnostische display van 16 hex karakters RFREQ, 260 keer per seconde gebeurt. Zou je sneller aan de afstemknop draaien (meer dan 2,5 omwenteling per seconde), dan nog zou dat geen probleem geven, want de frequentie wordt bij elke knopstap altijd

onder interrupt met prioriteit correct bijgewerkt in het RAM geheugen, zodat bij de eerstvolgende rekenlus de laatst ingestelde frequentie wordt gebruikt. Er kan dus geen slip optreden bij te snel knopdraaien tengevolge van de berekening en de instelling van de Si570 met de rekenresultaten.

Dat gereken kost dus minder dan 4 ms en dat terwijl de Si570 zelf volgens de specs tot 10 ms nodig kan hebben om een wijziging te effectueren in dat geval.

Nu we toch aan het meten zijn, even gemeten hoe lang het duurt om alleen de display vol te schrijven met 2 regels van 16 karakters elk; (diagnostic routine test_LCD daarvoor geschreven) het blijkt dat dat 452 keer per seconde gebeurt, dus pakweg 2 ms voor een volledig beeld en 1 ms voor alleen de volledige frequentieregel van 16 posities. Met een optische encoder van 100 stappen per omwenteling moet je dus meer dan 5 keer per seconde de knop 360 graden verdraaien wil de frequentiedisplay dat niet bij kunnen benen, terwijl er toch geen slip optreedt. Mijn ogen zijn een stuk trager, maar ja die kun je niet als referentie nemen want die kunnen alleen nog maar diep in een glaasje kijken teneinde SMD montage mogelijk te maken.

Alle tabellen zijn geprogrammeerd in C, in een programma dat de tabellen als assemblerlisting aflevert. Je kunt namelijk geen fouten toelaten in die tabellen, die er toe zouden leiden dat op sommige frequenties de VFO heel iets anders afgeeft dan de schaal aangeeft. Bovendien spaart het veel intype en uitzoekwerk uit.

Details Calibratie

Nu eerst eens nader kijken naar de calibratie. De bedoeling is de VFO met zijn output zero beat te zetten met een 10

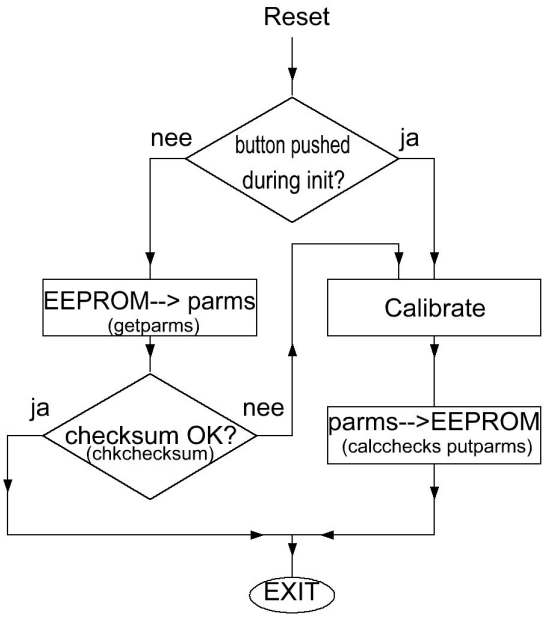


fig 6 EEPROM en checksum

MHz ijksignaal, WWV of een goede gecalibreerde teller, maakt niet uit. De VFO komt als je tijdens inschakelen van de netspanning de bedieningsknop al ingedrukt hebt, in de calibratiestand. Dat wordt door tekst op de tweede displayregel aangegeven. Dat doet hij trouwens ook als de checksum van alle gegevens in de EEPROM niet meer zou kloppen bij inschakelen van de VFO. Dat is bijvoorbeeld het geval als je hem na de bouw voor de eerste keer inschakelt en verder zo ongeveer een keer per twee jaar om onduidelijke reden. Een euvel dat je vaker hoort bij EEPROM's van controllers. Fig 6 geeft een toelichting.

Je kunt de frequentie die op de LCD precies 10 MHz aanwijst, terwijl de VFO dan in de buurt van 10 MHz staat, verdraaien tot er precies 10 MHz uitkomt, de range is maximaal 20 kHz naar boven en naar beneden, omdat de fabrikant dat als kristaltolerantie opgeeft. Of je snel of langzaam aan de actuator draait maakt bij de calibratie nauwelijks uit, die is dan altijd 10 Hz of 1 Hz per stap. De schaal wijst, als de output (na een opwarmperiode) zero beat is met je ijksignaal, dan dus inmiddels wat anders aan, dat in de buurt ligt van 10 MHz, omdat de kristallen per Si570 verschillen. De limiet voor verdraaien van plus en min 20 kHz, helpt trouwens ook om te voorkomen dat de voor 10 MHz ingestelde deeltallen N1 en HS_DIV niet meer zouden kloppen. Als de output precies zero beat staat met een ijksignaal van 10 MHz, druk je weer op de knop.

De 1/Xtal waarde voor de gebruikte Si570 wordt dan als reactie daarop bepaald uit

1/Xtal=RFREQ/(10^7*N1*HS_DIV)

Nu is N1*HS_DIV bekend. Omdat 10 MHz in hex notatie 00986080 is en de gebruikte index in de delerbepalende tabellen dus 38. Daaruit volgt dat bij 10 MHz die delers op 6*82=492 staan. De RFREQ in de teller wordt bepaald door de Si570 uit te lezen, terwijl die 10 MHz afgeeft. Die 2 getallen worden dan gedeeld in de 4byte/4byte deelroutine, waarmede als quotient 1/Xtal is bepaald. Dat wordt in de 96 byte RAM parameterbuffer gezet ter direct opvolgend gebruik van de VFO en die RAM buffer wordt tevens in EEPROM met met een erover berekende checksum gezet, zodat we later bij inschakelen en laden uit de EEPROM kunnen controleren of er geen falsificaties ingeslopen zijn. Mocht dat het geval zijn dan gaat bij inschakelen de VFO automatisch over in de calibratiemode. Na laden uit de EEPROM van de parameterbuffer en eventueel benodigde calibratie, gaat de VFO over in de normale mode.

Die deelroutine is voorshands alleen maar nodig voor de calibratie en dan zijn er geen vermenigvuldigingen, zodat daarvoor dezelfde RAM bytes als voor de vermenigvuldiging kunnen worden gebruikt. "Us bin sunig", maar dat wisten de Belgen al wel.

Bandgrenzen

Elke wijziging van de displayfrequentie, gebeurt onder interrupt. Het hoofdprogramma zou dan net bezig kunnen zijn

de binaire frequentie te wijzigen, om dat te voorkomen, wordt de displayfrequentie gekopieerd in een 9 bytes buffer alvorens er mee te gaan rekenen, en tijdens dat kopieren is de interruptroutine van de actuator even afgezet.

Bij elke berekening van de binaire frequentie wordt eerst gekeken of de BCD frequentie op de display niet buiten de bandgrenzen valt. Die zijn in ieder geval 3445 kHz en 160 MHz voor de CMOS uitvoering, maar bij de amateurbandfrequenties wordt binnen die banden gebleven als je de band met een optionele duimwielschakelaar op 4 ingangspoten van de controller P0.0 t/m P0.3 kiest. Zo'n schakelaar kiest 0 t/m 9 dus maximaal 10 banden, gebruik je jumpers of gewone tumblerschakelaartjes met waarde 1,2,4 en 8, of een hexadecimale duimwielschakelaar die van 0 t/m F loopt, dan kun je tot 16 banden gaan. Bij omhoogdraaien van de frequentie weigert de frequentie boven de bovengrens te gaan en evenzo bij naar beneden draaien onder de onderbandgrens. De bandgrenzen zijn voor 16 banden als 9 byte BCD waarden in het programma opgenomen 13 banden van 80 meter t/m 70 cm, en een breedband versie van 3445 kHz tot wat je IC haalt, bij mij dus 160 MHz, en nog een paar reservebanden. Zou de bandschakelaar bij calibratie niet op een band staan die 10 MHz bevat, dan wordt daar tijdens de calibratie, geen acht op geslagen. Onafhankelijk van de gekozen band komt de LCD op 10 MHz te staan.

De EEPROM

De controller bevat een EEPROM, programmeren kan per byte, kost 4 ms per byte, of ook per page van 32 bytes, dat duurt evenlang. Daarvoor is dus gekozen; het uitlezen gaat veel sneller en kan alleen per byte.

De EEPROM is min of meer beveiligd tegen schrijven, en hoewel ik de beveiliging optimaal gebruik volgens voorschrift fabrikant, komt het bij andere ontwerpen, die ik dagelijks in gebruik heb (Kujer2 bijvoorbeeld) toch ongeveer een keer per 2 jaar voor dat de EEPROM niet afgeeft wat er in gezet is. Meestal na een fikse onweersbui. Dat constateer ik doordat er een checksum over de aangeboden inhoud ingeprogrammeerd wordt en die klopt dan heel soms niet bij uitlezen. Vandaar dat ik bij opbergen van gewijzigde paramaterbuffer in RAM, daarvan een checksum bereken, en die tezamen met de bufferdata in de grootte van gehele pages (veelvouden van 32 bytes) in de EEPROM programmeer. De checksom die in het eerste byte wordt gezet is de som zonder carry van alle overige bytes van de buffer, (ofwel anders geformuleerd: de rest van de deling van de som van die bytes door 256), met een minteken ervoorgezet. Bij laden uit de EEPROM moet de berekende checksum over de hele geladen RAM-buffer inclusief de geladen checksum dus 0 zijn. Lijkt wat ingewikkeld misschien, maar dat inverteren heb ik zo gedaan om te voorkomen dat een of andere default inhoud, zoals alle bytes 00, een correcte checksum zouden opleveren.

Dat zaakje is getest, door de RAM buffer van getallen 1 t/m 63 te voorzien, de checksum daarvan te berekenen en in byte 0 te zetten, vervolgens te programmeren in EEPROM, de RAM-buffer te wissen, de buffer in te lezen uit EEPROM, en de eerste 8 bytes waaronder de checksum op de LCD diplay te schrijven. De checksum is ter controle zelf te berekenen met de van de ULO onthouden formule voor de som van een rekenkundige reeks, en die klopt (na een bug verwijderd te hebben).

Bij elk ontwerp is controle van jezelf bij elke stap dus belangrijk. In feite verschilt dat niet van het bouwen van een kitje, waar je bij elke stap moet controleren wat je deed. Juiste weerstandwaarde, IC niet verkeerd om, elco niet verkeerd om, soldeerpunten controleren. Op elk niveau is het controleren van jezelf uitermate belangrijk.

Frequentiegeheugen

Het is lastig, als je de zaak inschakelt dat je iedere keer een (huis)frequentie moet opzoeken. Daarom is een frequentiegeheugen ingebouwd. Druk je tijdens de werking van de VFO op de knop, dan wordt de huidige frequentie op de LCDisplay in het EEPROM geplaatst. Schakel je dan later de VFO weer in op die band dan komt hij onmiddellijk op die frequentie uit. Is voor die band geen frequentie bewaard, dan pakt hij altijd de lage zijde van die gekozen band als startfrequentie, omdat dat het CW bandbegin is. Totaal zijn er 16 mogelijke frequenties, een per band. Schakel je dus van band dan krijg je daar je voorkeursfrequentie direct te zien. Is die er niet, dan de ondergrens van de ingeschakelde band.

Nu is de RAM buffer in de controller uitgedrukt in toegelaten pages van de EEPROM van 32 bytes, beperkt tot 96 bytes (3 pages) voor dit doel. Daar zit al de checksum bij en de 1/Xtal 4 bytes en een MF offset, zodat er 87 overblijven. Daar kun je maar 9 frequenties van 9 bytes in kwijt en geen 16. Vandaar dat de frequenties bij opbergen packed BCD worden gecodeerd in 5 bytes, en bij uitlezen dus weer gedecodeerd in gewoon BCD. Dat levert dan een reserve op van 7 bytes in de RAM parameter buffer.

De 3500 ppm grenzen

iedere keer dat een nieuwe frequentie wordt ingesteld wordt gekeken of dit "on the fly" kan omdat de afwijking van de laatste centraalinstelling minder dan 3500 ppm is of niet. Bij een centraalinstelling worden daarvan enkele bytes pfn in het geheugen bewaard. Bij elke nieuw berekende RFREQ wordt gekeken of die niet meer dan 0x0.26 afwijkt dan die centraalinstelling, wat aan de krappe kant worse case overeenkomt met 3500 ppm. Zo ja dan kan on the fly de RFREQ worden gewijzigd, zo niet dan wordt de nieuwe frequentie als nieuwe centraalfrequentie gekozen en de pfn bytes, die de laatste centraalwaarde aabgeven, bijgewerkt naar de nieuwe waarde.

SDR

Bij software defined radio experimenteren is een VFO vereist die twee 90 graden in fase verschoven signalen afgeeft. Ook leuk trouwens voor een Fase-SSB zender. Dat wordt normaal gedaan met een dubbele D flipflop geschakeld als Johnson counter. De VFO moet dan 4 keer de op de schaal afgelezen frequentie afgeven, Uiteraard betekent dat, dat je niet hoger dan 40 MHz op de schaal kunt krijgen als je VFO tot 160 MHz gaat.

MF

Als je een tranceiver maakt is de ontvangstfrequentie oscillator minus de middenfrequentie MF, daarom is P3.6 een geschakelde input. Is die hoog dat is de zender actief en als die laag is wordt de middenfrequentie opgeteld om de VFO als ontvangstoscillator voor de mixer te gebruiken. Je komt dus op een (mogelijk aanzienlijk) hogere werkelijke oscillatorfrequentie dan de display aangeeft. Daarom wordt gecontroleerd of die de maximaal toelaatbare waarde van de Si570 (bij mij 160 MHz) niet overschrijdt. Zo wel dan wordt de Si570 niet opnieuw ingesteld maar verschijnt er een overflow waarschuwing op de display in plaats van de ontvangstfrequentie.

Voorzichtigheid is de moeder van de porseleinkast

Dit verhaal lezende zal duidelijk zijn dat ik voorzichtig te werk ga, teneinde geen onderdelen op te blazen en ik me tevens aan de fabrieksspecificaties houd. De fabriek heeft er immers geen belang bij slechtere specs te publiceren dan ze waar kunnen maken. Je weet dan ook zeker dat de zaak reproduceerbaar nagebouwd kan worden.

Wat wil echter het geval: Een analoge stroommeter van een half ampere volle schaal, wijst ongeveer 100 mA aan als gebruiksstroom van de Si570, ik geef en zwengel aan de actuator en dan blijkt soms de stroom omhoog te zwiepen tot 300 a 500 mA. In paniek de stroom verbroken, en de Si570 heeft het blijkbaar overleefd, zover ik dat kan vaststellen.

Als er wat fout gaat in de software zoals RFREQ buiten zijn grenzen, of een ack ontbreekt van een der I2C stuurroutines dan wordt dat gemeld met een foutbericht op de onderste regel van de LCD. Het blijkt nu dat die rare verschijnselen soms optreden, als de Si570 een nieuwe verwegfrequentieinstelling krijgt, waardoor hij een tijdje geen output geeft, volgens de fabrikant maximaal 10 ms niet. Intussen loopt het programma wel door en als die dan een nieuwe instelling aanbiedt via de I2C aan de Si570 voordat die weer output geeft, dan is het hommeles, althans af en toe. Dat is dan een slecht ontwerp van Siliconlabs want van de I2C bus kan de klok laag gehouden worden door de slave, (klok stretching) als hij nog niet klaar is, en dat gebeurt blijkbaar, als voorlopige conclusie, niet.

Ik heb de zaak opgelost door bij een nieuwe centraalinstelling van de Si570 een downcounter te starten van 10 ms die in de timer_0 interrupt verlaagd wordt tot die uiteindelijk na 10 ms 0 is geworden. Komt het main-programma dan voor een nieuwe instelling bij de I2C bus voor die 10 ms om zijn, dan wordt slechts gewacht tot ze wel om zijn, als er inmiddels geen nieuwe actuatorstand is aangeboden. Is dat wel zo dan wordt de Si570 niet ingesteld maar wordt de nieuwe stand berekend en onder dezelfde condities aangeboden aan de I2C bus.

De display

De display duurt ongeveer 2 ms om die volledig te vullen. Dat is bijna net zo lang als de hele rekenloop erover doet om een volgende instelling van de Si570 te berekenen. Een deel van die display-tijd is wachtverlies, omdat na het laden van een karakter hij wacht tot een busyvlag op vrij komt te staan. Voorts wordt bij voortdurend draaien aan de actuatorknop, een paar honderd keer per seconde de main loop doorlopen en dus ook steeds de display ververst, terwijl we helemaal niet zo snel kunnen lezen, en de LCD kristallen ook niet supersnel zijn, vooral niet als het koud in huis is om uit de brandstofbesparing onze hobby uitgaven te kunnen betalen. Verversen 25 keer per seconde zou genoeg zijn, dus daar is op twee fronten tijdwinstwinst te halen. Nodig of niet, maakt niet uit.

Dat kan gebeuren door de displayoutput niet in de display maar in het geheugen te schrijven. Op interruptbasis (timer_1 overflow) wordt om de milliseconde er een karakter uitgehaald en in de display gezet. De karakterspatie in de tijd is dan zo groot dat niet op de busy vlag gewacht hoeft te worden. De 2 lijnen worden om de beurt afgehandeld, het kan dus niet zo zijn dat als de eerste lijn steeds ververst wordt de tweede niet aan de beurt komt. Een lijn die niet gewijzigd is wordt niet opnieuw op de display gezet. Als een lijn na 16 interrupts is afgehandeld, wordt de cursorteller weer op de linkerzijde van de display gezet en wordt _een

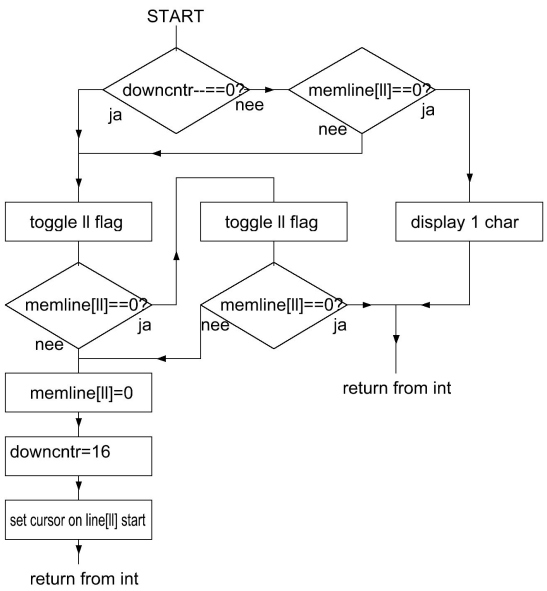


fig 8 Flow chart van display interrupt afhandeling

interruptafhandeling gebruikt om de cursor van de display op de juiste lijn te zetten. Is er geen wijziging in de display-lijn-RAMbuffers, dan gebeurt er niets. Wordt tijdens de display van een lijn de lijn inmiddels ververst in het dispay-geheugen door de rekenloop, dan begint de eerstevolgende interrupt weer aan het begin van de lijn. Dit alles wordt voorelkaar gekregen met een paar vlaggen die de status weergeven, en doordat het hoofdprogramma per lijn een vlag memline[2] zet, die bij de eerstvolgende interruptafhandeling van de betreffende displaylijn gereset wordt. In fig 8 wordt de werkwijze van de interruptroutine nader toegelicht. Uitgewerkt plan, echter niet geimplementeerd.

CW en MF offset

Als je de VFO wilt gebruiken in een tranceiver kun je niet op dezelfde vfo frequentie zenden als ontvangen want dan is de CW zero beat. Daarom is een pen gereserveerd, die hoog is bij zenden (default) de schaalwaarde afgeeft aan signaal en bij ontvangst (pen laag) een hoger in frequentie signaal. De bedoeling is dat de offset die hier zowel de middenfrequentie (0 bij DC ontvangers) als het verschil met zero beat omhelst, precies centraal in je doorlaatband van je filter valt. Daarom is die offset regelbaar gemaakt en kan tijdens de calibratiefase worden ingesteld.

Twee VFO's A en B

Je kunt de zaak steeds uitgebreider maken (fig 7) , zo leek me wel aantrekkelijk 2 vfo's te hebben die omschakelen van de een naar de ander bij overgang van zenden op ontvangen. Dan immers kun je met ongewijzigd vasthouden van je ontvangstfrequentie, onafhankelijk daarvan 1 kHz up of iets dergelijks om te zenden. Dat vinden pile-up hams aantrekkelijk. Ik heb geen verstand van pile-ups, omdat ik de mij

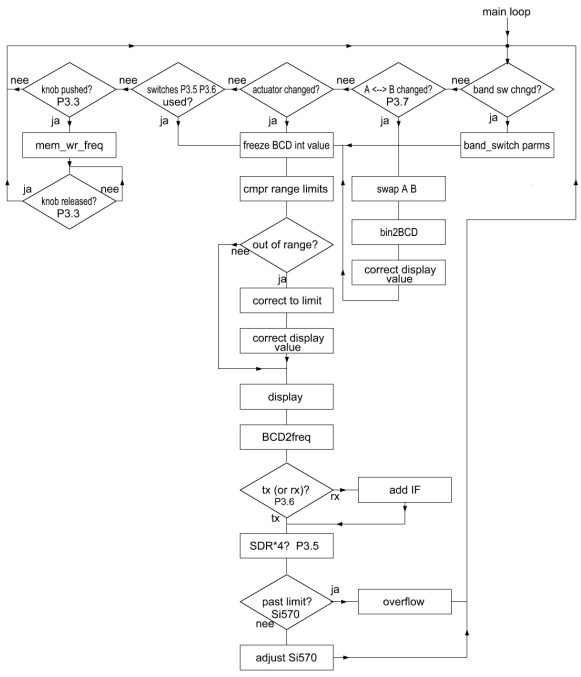


fig 7 Aparte VFO voor zenden en/of ontvangen (A<->B)

toegemeten tijd anders gebruik. In feite ontstaan er 4 mogelijkheden:

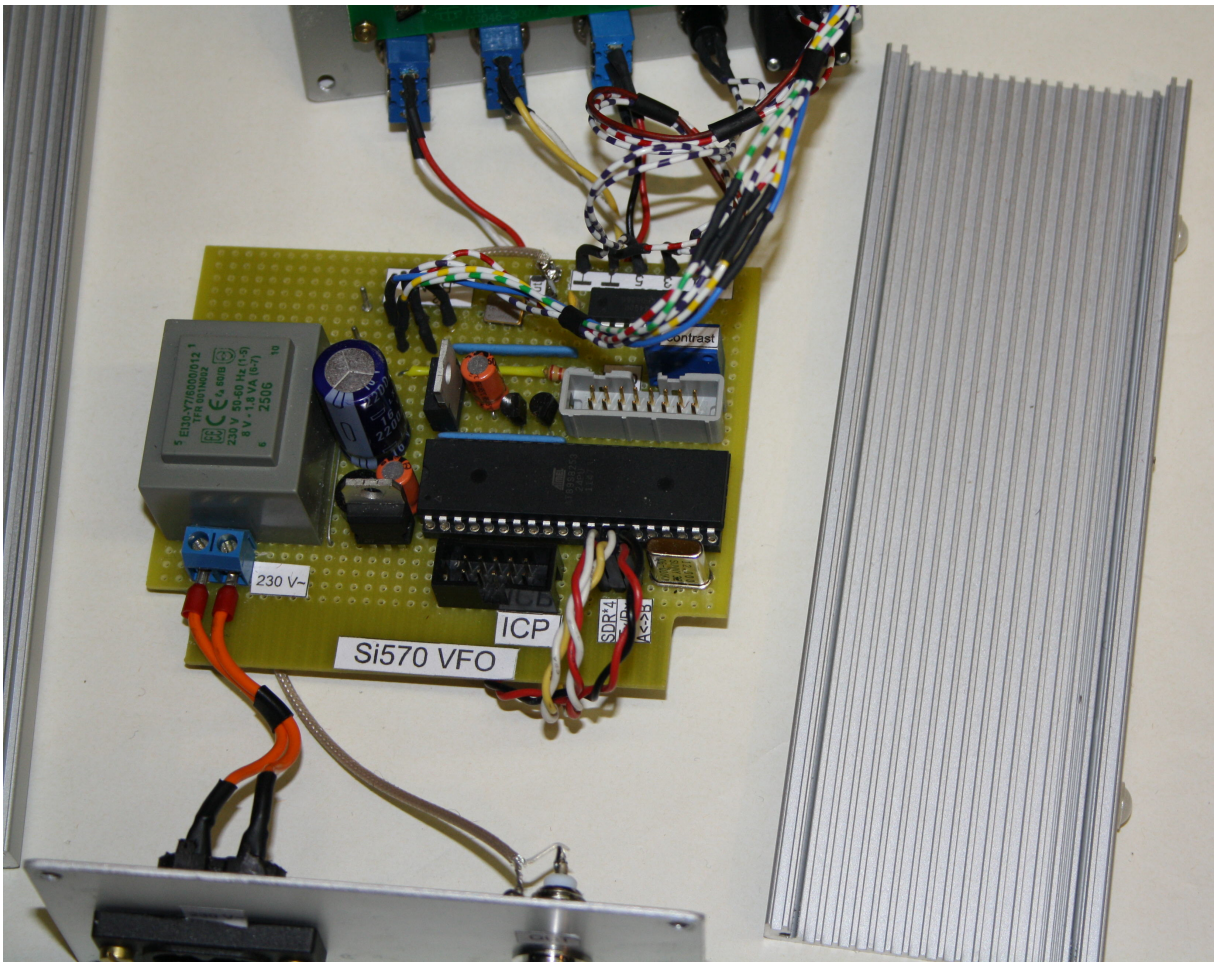
zenden op A, ontvangen op A
zenden op A, ontvangen op B
zenden op B, ontvangen op A
zenden op B, ontvangen op B

Daarvoor is een tweede 4 byte frequentiebuffer nodig, die dan omgewisseld kan worden met de eerste door een input-pen te bedienen. Maar de display van die tweede buffer is in 9 byte BCD, waar hij immers uit berekend werd. Die display moet nu dus (wegens dreigend RAM gebrek voor dit soort uitbreidingen) teruggerekend worden uit de omgewisselde binaire frequentiewaarde. Dat eist een hele rekenpartij, namelijk het 4 byte getal door 10 delen, de rest is de BCD digit, het quotient weer door 10 delen levert als rest het tweede BCD digit, dat door 10 delen moet dus 9 keer gebeuren voor 9 cijfers.

Eerder schreef ik dat de deelroutine binair bitsgewijze was gemaakt omdat hij toch maar eenmalig nodig was voor calibratie, en ik dat makkelijker vond te programmeren, nu echter noodde dit drieste plan me om de machineinstructie divAB te gaan gebruiken, die deelt byte in accu A door B en het resultaat (quotient) staat dan in A en de rest in B. Echt vlot gaat het niet want na berekenen van de rest moet je een volgende waarde aanhalen, net als bij staartdelen, en dat kan geen byte zijn omdat de rest al een nibble (4 bits) is dus je kunt slechts een nibble aanhalen om samen met de rest als een byte in accu A te zetten. Kortom een heel gedoe, lastig te programmeren, maar het is verder prima en compact gelukt, edoch heb ik geen moeite gedaan het verder te optimaliseren. Je hebt alsnog 8 divAB instructies per tiendeling dus totaal wordt 72 keer de divAB instructie uitgevoerd bij de terugtransformatie naar 9 BCD cijfers.

Je vraagt je dan af of er tijdwinst is tov de binaire bitsgewijze deling en zo ja, hoeveel. Dat heb ik uitgezocht. De terugtransformatie van 32 bits naar 9 BCD met de zojuist besproken voor dit doel geschreven routine kost 680 microseconde. Doen we het met de binaire bitsgewijze deler dan kost het per deling door 10 1,5 ms; en omdat er dan 9 delingen nodig zijn totaal 13,5 ms, aangevuld met tijd voor laden en lossen. De tijdwinst door gebruik van de divAB instructie is dus aanzienlijk.

Impementatie VFO-A en VFO-B De wijze waarop ik dat het best kon implemeteren heb ik overleg over gepleegd met Steef PA3S een bekend contest station, Henk PA1A is ook bekend, Freud zou kunnen verklaren waarom, maar die zijn zelf ontworpen mast is recent omgewaaid, kan natuurlijk, maar hij kraakte op zendamateur.com andere ontwerpers van masten af dus dan moet je moedertje natuur niet de kans geven je vermeende superioriteit te loochenstraffen; die laat ik dus maar even met rust. Steef PA3S schreef dat hij niet split werkt, en zijn response op mijn vragenstellerij was, zeker bruikbaar en geimplementeerd. Aldus heb ik besloten twee schakelpunten te implementeren, namelijk zenden/ontvangen en VFO A of B. Daar kun je alle kanten



mee op. Desnoods middels een paar nor of nand poortjes, zodat je niet voor niks de digitale poorten voor het F(ull) "examen" hebt geleerd. Het geleerde nog even toepassen, dus.

Metingen

Meting van tijdsduur van de actuatorintterruptafhandeling zijn gebeurd door de interrupt 0 pen te koppelen aan een pen die in het hoofdprogranmma steeds laag en weer hoog gezet wordt, zodat het hoofdprogramma niets anders doet dan interrupts opwekken. De interruptafhandelingsroutine keert steeds een outputpen van niveau om, en aldus kan met een teller op die pen de frequentie van interruptafhandelingen worden gemeten als de processor 100% wordt bezet ermee. Dat blijken er 22500 per seconde te zijn.

Meting van tijdsduur van een correctie en instelloop in het hoofdprogramma. Om dit te meten wacht het programma niet op de vlag 'changed', die op een interruptafhandeling wordt geset, maar wordt die gelijk doorgeschakeld als ware het dat de frequentie gewijzigd is. De display wordt bijgewerkt en de hele mainloop wordt doorlopen inclusief het bijwerken van de Si570. Dat gebeurt bijkens de meting 340 keer per seconde.

Als de zaak gewoon op de plank instaat, verloopt die volgens de aangesloten teller ongeveer top top 1,5 Hz op 3445 kHz. Wel met je vingers van het Si570 IC afblijven want dat verloopt met de temperatuur, die in de shack goed constant is, zolang we het aardgas voor verwarming nog kunnen betalen, en het verdronken land van Grunninghe nog niet op de kaart staat. Bij inschakelen verloopt de zaak wel, maar dat is na een kwartier of zo stabiel. Calibreren dus pas na verloop van die inschakeltijd doen.

Nabouw

Om plat trappen van een zelfbouwproject dat op de grond ligt (dan kan het niet vallen) te voorkomen, heb ik het in een kassie-6 gezet, Bestelnummer bij Conrad 523232. Kijk ook even bij Reichelt die leveren ze voor de helft van de prijs. Met een printer de boormalletjes fig 9 afdrukken op een grootte die past op frontje en achterkant. Uitknippen, op het aluminium tepen. Op de tekening met een centerpunt doortikken, boren met 2,5 mm, spiritus uitsluitend als smeermiddel gebruiken, niet van snoepen. Vervolgens naboren op de gewenste aangegeven afmetingen en rechthoekige gaten uitzagen met een figuurzaag. Gaten afbramen hen een grotere boor. Figuurzaagsneden uitvijlen met een plaatte bastaard tot op de kraslijn tekenrand, daarna afschuren met schuurpapier op een verfroerhoutje. Geprogrammeerde processor kost 16 euro inclusief verpakking en porto. Neem contact op met mijncall@amsat.org, waarbij mijncall uiteraard vervangen moet worden door

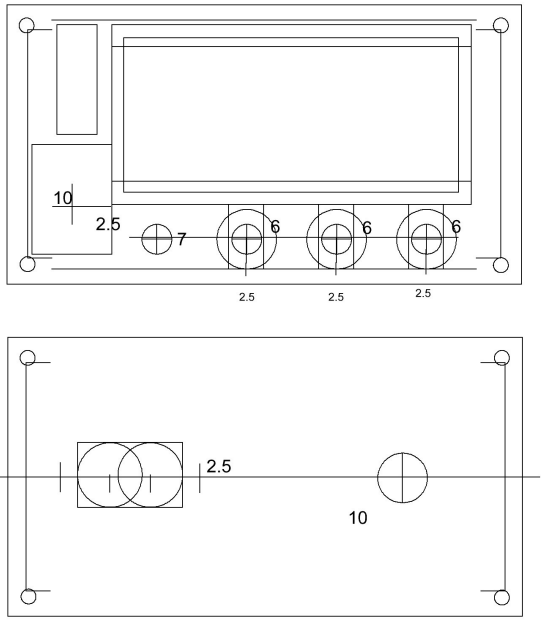


fig 9 Boorplan

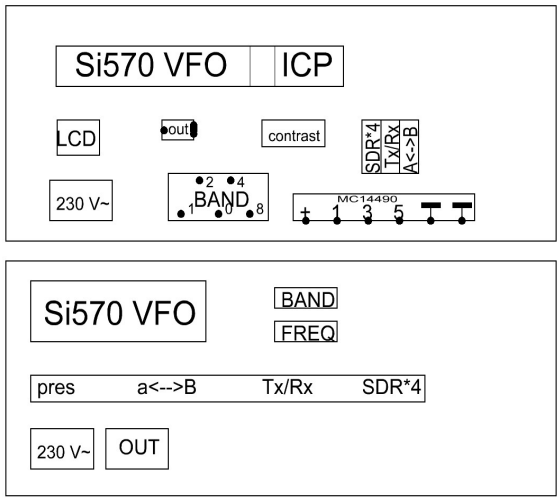


fig 10 Etiketten

PA0WV