Redactie Johan Smet ON5EX on5ex@uba.be
Rik Strobbe ON7YD on7yd@uba.be
Jos Warnier ON6WJ on6wj@skynet.be

Technical Info

Un double VFO Si570 (2ème partie)
Een dual Si570 VFO (deel 2)

Door/par PA0WV – Traduit par/vertaling ON7CFI

L'actuateur

C'est l'intention de pouvoir changer la fréquence par un bouton. Cela peut se faire moyennant un actuateur, c'est un interrupteur qui, en tournant le bouton, s'ouvre et se ferme continuellement, et ce, ensemble avec un deuxième interrupteur qui fait de même mais à 90°, ce qui permet de déterminer si le bouton tourne à droite ou à gauche. Ces interrupteurs rebondissent ("bounce") et c'est dérangeant, mais Bournes, le fabricant du type ECW1J-B24-AC0024L, spécifie qu'on peut utiliser un IC dé-bondisseur de Motorola, le MC14490P. Il est aussi possible de construire son propre actuateur optique en faisant tourner un disque avec des traits noirs et transparents sur un cercle dans un IC comme on peut en trouver dans une ancienne souris optique. Je ne l'ai pas encore fait, donc provisoirement, j'utilise un actuateur et un IC dé-bondisseur. La troisième possibilité consiste à acheter chez Farnell un "optical encoder", type ENA1J-B28-L00100L. Celui-ci fait 100 pas par tour et ne nécessite pas de dé-bondisseur car il ne bondi pas, mais il requière bien 25 mA 5 V d'alimentation. J'en ai acheté un et il fonctionne également. Le MC14490 n'est alors pas nécessaire, mais je l'ai gardé; on peut l'utiliser pour dé-bondir un petit bouton pressoir, bien que j'aie pu le faire dans le software, comme a été fait avec le commutateur de bande, voir plus loin.

Une patte de l'actuateur dé-bondi provoque une interruption externe sur P3.2. Quand celle-ci arrive, la routine de traitement des interruptions vérifie P3.4 pour déterminer si cette patte est basse ou haute, ce qui donne le sens de la rotation. A droite, la combinaison de la fréquence en 9 chiffres BCD – visible sur le display – est incrémentée, sinon elle est décrémentée. Le bout de programme écrit pour cela peut être vite testé. Pas tout fonctionne bien du premier coup, mais en regardant le display, on peut déduire ce qui cloche et ensuite le corriger.

Il est vrai que le VFO n'est pas stable à 1 Hz près (ne diverge pas beaucoup d'ailleurs), cependant, l'indication a été faite avec une résolution de 1 Hz, afin de pouvoir incrémenter ou décrémenter la fréquence de cette quantité minimale en tournant le bouton. Quand le "rate multiplier" a 28 bits après la virgule, cela donne une différence minimale de 2^{-28} *114 MHz mesuré au VCO, donc 0,425 Hz en fréquence VCO ce qui correspond à peu près à 1E-4 Hz per MHz.

Régler un VFO par petits pas de 1 Hz est intéressant quand on accorde un signal, mais pas si l'on veut changer de fréquence d'un bon coup. A 24 kHz on peut brancher une foreuse sur le bouton de réglage pour atteindre la nouvelle QRG. 24 Hz par tour du bouton de la version mécanique fait 1000 tours, en utilisant le "optical encoder" mentionné à 240 tours. Aussi différents tailles de pas sont désirables. La taille du pas est stocké dans un byte nommé delta dans le contrôleur, et celui-là contient quelque part un seul bit 1, Les autres bits restent 0; et la position de ce 1 spécifie la taille du pas; tout-à-fait à droite 1 Hz, tout-à-fait à gauche 10 MHz. A chaque position plus à gauche la taille du pas augmente d'un facteur 10. Pendant les tests, j'ai limité cela à maximalement 1 MHz, car à 100 pas par tour, on peut déjà changer

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 5 V 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 te 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 negencijferige 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 één en slechts één 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

de 100 MHz par tour en tournant vite. Quand on tourne lentement, la fréquence doit changer par Hz et au fur et à mesure qu'on tourne plus vite, la taille du pas doit augmenter.

Cela peut se faire en utilisant une interruption du timer_0 qui incrémente un compteur de 8192 pas. Compteur_0 s'incrémente à chaque overflow du compteur ce qui génère une interruption (244 par seconde) et incrémente alors le compteur de temps d'un byte en RAM. De l'actuateur on observe dans sa routine de traitement des interruptions la valeur qu'a atteint ce compteur de temps et on le remet à zéro. Si cette valeur est petite, on tourne visiblement vite et le 1 en delta est mis en position à gauche. Si la valeur est importante, donc quand il y a plus de temps entre deux impulsions de l'actuateur, le 1 en delta est mis à une position plus basse. Quand le compteur atteint 255, il n'incrémente plus mais reste à cette valeur jusqu'à ce que l'interruption de l'actuateur vienne, ce qui le remet à zéro.

Conversion vers le binaire

Il faut maintenant faire des calculs sur cette fréquence BCD en mémoire et sur le display. Pour cela une représentation en binaire est souhaitée. Après chaque modification du LCD, ce calcul est effectué. Cela se fait en mettant les montants 1, 10, 100... 10^8 en binaire dans la mémoire du programme, en multipliant ces montants par la valeur du BCD correspondante et en additionnant le tout. Etant donné que dans le contrôleur il y a bien de la mémoire en stock, mais que la vitesse de calcul n'est pas fulgurante, la multiplication peut être évitée en mettant les valeurs binaires des multiplications susmentionnées dans une table; de sorte que des additions de valeurs à 4 bytes venant de cette table restent à faire. La table a été écrite en langage assembler par un programme C écrit à cette fin, nommé multitab.c, et disponible sur mon site web sous le chapitre construction Si570_VFO. Cela évitera des erreurs de calcul, de frappe et d'autres erreurs et prévient le VFO de ne pas générer la fréquence indiquée au display. Surtout dans un émetteur où cela est important.

La fréquence maximale du Si570 en version CMOS est de 160 MHz, et cela doit être noté en binaire en 28 bits. Un byte vaut 8 bits, aussi notons les valeurs en 4 bytes. Cela suffirait à bien 4 GHz, la limite de 945 MHz des autres types Si570 ne posera donc pas de problèmes.

On est arrivé au stade où la fréquence indiquée au LCD en tournant à droite ou à gauche, à chaque modification peut être convertie en un nombre binaire à 4 bytes.

L'interface du Si570

Le Si570 tourne à 3,3 V, un stabilisateur "low drop" LM1117 3.3 V supplémentaire pour l'alimentation est donc nécessaire (voir **figure 4**). Pas seulement l'alimentation, mais aussi l'interface bidirectionnelle du bus I2C vers le processeur doit être doté d'un "level shifter" bidirectionnel 3,3 <–> 5 V. Philips indique dans "Application note AN97055" comment cela peut être fait avec deux pièces BS170 N channel MOSFETs. On peut les acheter chez Conrad sous le n° 158950 et coutent 51 ct quand on

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 telleroverflow 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 actuatorclicks, 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

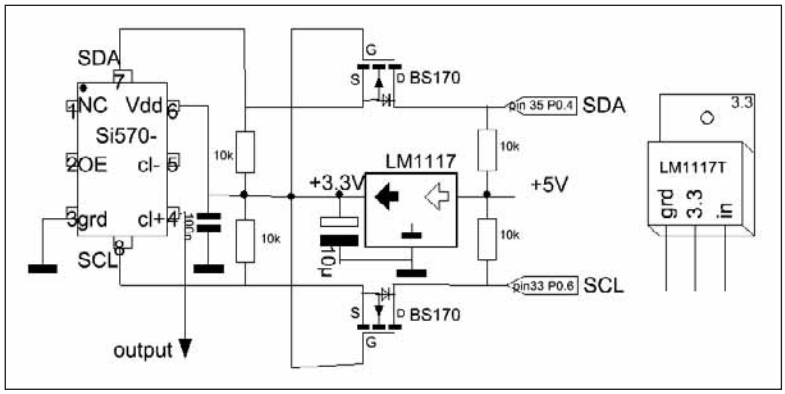
Nu moet er met die frequentie die in BCD in het geheugen en op de display staat, worden gerekend. Daarom is een binaire representatie gewenst. Na elke schaalwijziging op de LCD wordt die berekening uitgevoerd. Dat kan gebeuren door de getallen 1, 10, 100... 10^8 binair in het programmageheugen te zetten, die getallen te vermenigvuldigen met de bijbehorende BCD-waarde tussen 2 en 9 en dat hele zaakje op te tellen. Nu is 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, tikfouten en vergissingen, wat voorkomt 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 binair te noteren in 28 bits. Een byte is 8 bits dus we noteren de waarden in 4 bytes. Dat zou zelfs voldoende zijn voor ruim 4 GHz, zodat de 945 MHz limiet van de andere types 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 **figuur 4**). Niet alleen de voeding, maar ook de bidirectionele interface van de I2C-bus naar de processor moet 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



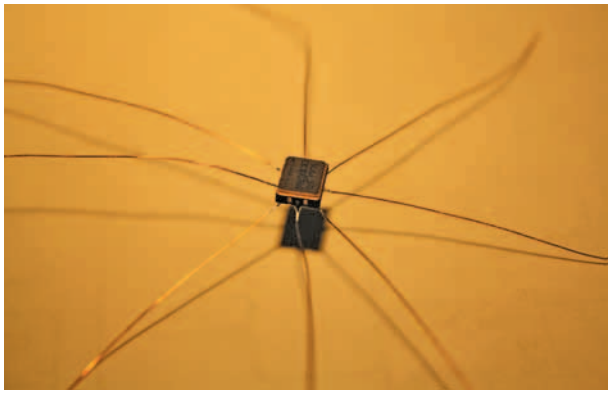


Fig. 4. Si570 alimentation et interface I2C / *Fig. 4. Si570 voeding en I2C interface*

Photo 3 / *Foto 3*

en achète 10 pièces, ce que j'ai fait. On peut les utiliser également en simple ou branchés en parallèle comme étage final QRP jusque 5 Watts.

Sur les îles à souder du Si570 on soude des bouts de fil d'alimentation souple dénudés. Ceux-là sont en cuivre et à peu près de 0,18 mm d'épaisseur. La **Photo 3** montre comment transformer une mouche en araignée. Faites attention aux connexions du BS170, elles différent selon les fabricants. Conrad m'a fourni des Fairchild et là, quand on tient les fils dans la direction de ses yeux la face aplatie du MOSFET en bas de gauche à droite c'est Source-Gate-Drain. Le drain doit être connecté au contrôleur.

Bref, toute sorte d'erreurs sont possibles, on monte donc d'abord l'alimentation 3,3 V et les "level shifters". Comme résistances 10 k pull-up j'ai utilisé des barrettes à huit résistances 10 k la pièce. La disposition de ces composants sur le circuit perforé est dessinée séparément en **figure 5**. Toutes les pattes du port P0 sont connectées via un tel pull up au +5, nécessaire afin de pouvoir utiliser les autres pattes de P0 à d'autres fins.

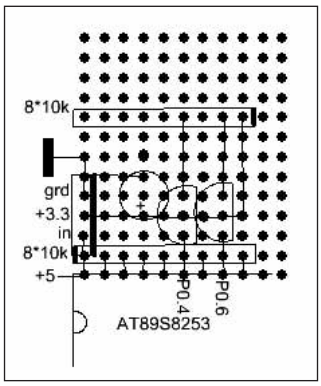


Fig. 5. Face composants montage Si570 interface.

Fig. 5. Onderdelenzijde montage Si570 interface.

Mesurez pour vérifier les tensions, ensuite une routine de test a été faite qui commute une patte du contrôleur et vérifiez si les autres pattes du contrôleur sont configurées comme entrées et suivent le rythme et également commutées sont les entrées et les sorties. Evidemment pour cet essai, le SCL et le SDA du côté Si570 des "level shifters" sont branchés l'un sur l' autre. La routine fait partie du diagnostic du programme sous le nom test_I2C. Le Si570 sera branché seulement après ces contrôles. Il n'y a en fait que 5 fils à connecter: SDA, SCL, grd, Vdd et output. OE ne doit pas être connecté, il est activé en interne, et une patte n'est pas connectée (NC). Ce modèle n'a pas de sortie balancée Cl-, ce qui laisse 5 connections. L'alimentation 3,3 V est découplée près du Si570 avec un C 100 nF céramique. Les fils soudés en excès peuvent être enlevés.

Brancher la tension et sortir le compteur de fréquence, il sort presque 15 MHz, il y a de la vie là-dedans! Vérifier qu'il n'y a effectivement pas de sortie sur Cl-. La dérive de température après l'allumage est de 200 Hz vers le bas avant de se stabiliser à presque 1 Hz.

L'adresse I2C

L'étape suivante est de constater l'adresse du Si570, ce qui devrait correspondre à la deuxième ligne du n° type. Elle lit: CBC000266G, mais cette histoire ne m'est pas claire. Pour avancer le schmilblick, j'ai écrit suffisamment pour assembler et pour faire fonctionner le bus I2C en tant que maître. Les 128 adresses possibles sont tiraillées en rafale et on vérifie si une réponse vient du Si570 sous forme d'un bit ACK. Cette adresse est alors affichée au display, ainsi on le sait. Il s'avère que c'est hex 55, un chiffre que je ne reconnais pas dans le n° type.

Lire et écrire le Si570

Les routines pour lire et écrire une commande complète sont maintenant écrites, comme spécifié dans le datasheet du Si570 à la page 16. En guise de test la valeur des registres 7 à 12 du Si570 est lue et mis en hex au display. Après quelques débogages, cela fonctionne également. Dans mon cas, cela donne A8C2A85D74F3. Selon le specsheet cela veut dire: diviseur rapide HS_DIV= 9, second diviseur N1= 36. Le produit de ces deux fait 324, ce qui donne la plage de fréquence de sortie complète du VCO du Si570 une largeur de 14,969 à 17,5 MHz. La fréquence de sortie mesurée se situe dans cette plage, c'est donc plausible. Le RFREQ, c'est le diviseur du VCO qui donne la fréquence du cristal, est également apparent de l'affichage LCD. Il y a 28 bits après la virgule, le nombre est donc hex 2A,85D74F3. Mon compteur indique finalement 14,999880 MHz et une fluctuation de 1 hertz à long terme. De ça la fréquence du cristal du Si570 peut être calculée à la précision de mon compteur de fréquence, c.-à-d. 324 fois l'affichage du compteur donne

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. **Foto 3** 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, 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 **figuur 5**. Alle poten van port P0 zijn via zo'n pull up aan +5 gehangen, nodig om die andere poten van P0 ook te kunnen gebruiken voor andere doeleinden.

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

la fréquence du VCO et diviser celle-là par RFREQ donne la fréquence du cristal. Ce qui donne 114,29066 MHz ce qui est à nouveau plausible.

Comme le cristal doit résonner dans la plage du VCO, RFREQ doit être minimalement 42 en maximalement 49 devant la virgule, ou en hex 0x2A à 0x31. On a à faire à 2 diviseurs HS_DIV et N1 en cascade suivant le VCO, fournissant la fréquence de sortie. Les diviseurs doivent être choisis pour obtenir la fréquence de sortie voulue du VFO avec le VCO entre les bornes GHz susmentionnées. Pour cela, j'ai écrit un programme en C (vfomult.c) – la sortie est également disponible sur mon site web sous le nom delers.txt – qui contient toutes les combinaisons possibles du diviseur rapide et du diviseur lent avec une valeur minimale du produit égale à 6. Ce qui va jusque 945 MHz. En théorie on peut utiliser également 4 et 5 comme produit, mais on obtient alors des trous dans les bandes le plus hautes.

Il s'avère que pour l'IC CMOS utilisé pas tous les petits diviseurs sont utilisables car on obtient alors un produit de HS_DIV (le diviseur rapide) et N1 (le diviseur lent) en dessous de 32 et une fréquence de sortie au-dessus de 160 MHz, ce qui est la limite supérieure pour cette version CMOS. L'un peut avoir 6 valeurs admissibles, l'autre 65 valeurs différentes, soit en total 390 combinaisons. Ces combinaisons sont triées selon la valeur du produit qui est, jusque 945 MHz, entre 6 et 1408, et cela spécifie les plages de fréquence correspondant à ces combinaisons. Il y en a moins que 390, car un certain nombre des combinaisons ont le même produit. Ces doubles ayant le même produit ont été éliminés du programme vfomult.c, en gardant le produit ayant la valeur la plus haute du facteur HS_DIV, comme recommandé dans le data sheet Si570, ce qui laisse 276 combinaisons dans la liste delers.txt. La liste pourrait être nettoyée davantage comme on constate que toutes les plages de fréquence envoisinant se chevauchent plus ou moins. L'inspection plus poussée de la table delers.txt nous montre que seulement 34 pairs de diviseurs suffisent pour combler la plage complète de 3,5 à 945 MHz avec un chevauchement minimal. Cela ne doit pas nous étonner car le rapport de la fréquence maximale à la fréquence minimale du VCO = 5, 67/4,85=1,17 et le rapport de 945 MHz / (1,17 * 3445 kHz) = 235 ce qui fait que le nombre minimal théorique de pairs n égale $1{,}17^n$=235, où n est donc log(235)/log(1,17)= 35.

Les fréquences de sortie du VFO peuvent donc être obtenues avec du chevauchement. La "specsheet" conseille en cas de chevauchement l'utilisation de la fréquence VCO la plus basse (donc le diviseur N1*HSDIV le plus petit) parce qu'alors la dissipation (et donc la dérive) est un peu plus faible. La "specsheet" stipule encore que la fréquence minimale utilisable est de 10 MHz. Je ne sais pas pourquoi elle dit cela, car la fréquence est déterminée en mettant les deux diviseurs à leur valeur maximale. Je ne vois pas de bonne raison qui empêcherait cela. Fonctionner cela fait certainement. La fréquence minimale que produit le Si570 est donc 3445 kHz. Pour les appareils SDR qui aiment recevoir le quadruple de la fréquence nominale, on peut donc utiliser un quart de cette valeur, à cause du diviseur Johnson externe (divise par 4). Il ne faut donc pas rater les allocutions radicales de l'Imam sur 1007 kHz de la radiodiffusion publique néerlandaise quand on utilise ce VFO pour la réception SDR.

Tester le pilotage I2C

Tout ceci étant connu, la routine d'écriture I2C peut être testée en écrivant d'autres valeurs de 6 bytes dans les registres reg7 à reg12 du Si570 et de vérifier si la fréquence de sortie du VFO correspond. De plus, ces registres peuvent être lus à nouveau et affichés comme double vérification par la routine de lecture I2C.

Les valeurs peuvent être écrites, ce qui est démontré par la relecture susmentionnée. Pourtant, la fréquence à la sortie ne change pas en cas de changement important, parce que cela requière des signaux de contrôle supplémentaires qui doivent encore être programmés dans les registres reg135 et reg137 du Si570. Ce qui coute alors maximalement 10 ms d'interruption du signal de sortie selon le datasheet. Cela vaut pour chaque changement de RFREQ plus grnad que 3500 ppm. Un

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 buurentries 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 noodzakelijke 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 dubbelcheck 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

changement plus important requière une nouvelle valeur centrale de RFREQ et potentiellement des diviseurs, moyennent commande via les registres 135 et 137 du Si570.

On peut donc changer le VCO et donc RFREQ dans une tolérance de 3500 Hz par MHz au-dessus comme au-dessous, sans aucun autre changement que d'écrire (une partie de) RFREQ. RFREQ se situe entre 0x2A et 0x32 ce qui détermine les pas de 114 MHz de la fréquence du VCO. Les 28 bits après la virgule en déterminent une partie.

Les calculs sont fait en binaire. L'amplitude de la déviation permise de RFREQ pour 3500 ppm peut être calculée à minimalement 3500E-6*RFREQ_min = 0x0,2605B91. Ce qui veut dire qu'on peut changer RFREQ sans interruption de la sortie "on the fly" pour autant qu'on reste dans la déviation par rapport à la dernière valeur nominale écrite. Quand on dévie trop loin, le VCO perd la synchronisation. Sur un compteur de fréquence à la sortie du Si570 cela est clairement visible car la stabilité de roche du signal de sortie diminue sensiblement.

Des pas plus grands que 3500 ppm de la fréquence doivent être faits en arrêtant le VCO en écrivant le registre 137, les nouvelles valeurs des diviseurs et de RFREQ, pour autant qu'ils aient été changés, de démarrer le VFO à nouveau via le registre 137 et en activant ensuite le bit "newfreq" dans le registre 135. Quand on reste dans la plage avec les mêmes diviseurs, il suffit, parce que les diviseurs et RFREQ sont réglés correctement à la limite de la plage autorisée correct, d'arrêter le VFO, de le démarrer et de parcourir une procédure de "setfreq", puisqu'il n'y a pas de registres à changer.

Dissipation et dérive

Le "specsheet"recommande en cas de choix multiple de choisir HS_DIV le plus haut possible et donc N1 le plus petit possible pour obtenir le produit désiré HD_DIV*N1. Cela dissiperait moins et réduirait donc la variation de température dans le chip, ce qui améliore la stabilité de la sortie. Il est également avisé en cas de choix multiple (ce qui peut être contraire à l'exigence précédente) de maintenir le VCO dans la partie basse de la plage de fréquence, donc de choisir le plus petit produit HS_DIV * N1 possible pour obtenir la fréquence désiré.

Afin de se donner une impression, je mesure le courant consommé à la même valeur basse du VCO (= basse RFREQ=0x2B,0) pour les diviseurs 4 fois 22 et pour 11 fois 8, tous deux donc avec le même produit 88, l'un après l'autre. Cette mesure n'est pas nécessaire, mais on a obtenu sa licence pour faire de la recherche et de l'épanouissement personnel, pas pour des petits concours radio et du radotage dans un microphone d'une boîte japonaise aux réglages d'usine et une antenne déballée de son blister.

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

Il y quatre possibilités: VFO au-dessus et au-dessous dans sa plage, et les diviseurs comme décrit. Ce qui donne les résultats dans le tableau. Il s'avère des résultats mesurés que la fréquence calculée du cristal suite à la variation de température causée par la variation de la dissipation flotte de bien 40 Hz vers le haut et vers le bas. Laisser diviser par 88 les deux diviseurs en gardant le VCO au même endroit donne déjà une différence de 20 Hz, par la dissipation différente dans les diviseurs tandis que la fréquence devrait rester la même.

Avec le même diviseur, changer le VFO de gauche à droite dans sa plage, donne aussi une différence de dissipation et la fréquence calculée du cristal diffère donc ce qui montre la dérive du cristal, montrée dans le tableau. Le tableau a été généré en programmant le contrôleur tel qu'il choisisse cycliquement les fréquences pour bien 16 secondes chaque. Immédiatement après un changement la dérive est la plus importante. Les valeurs mentionnées sont les valeurs finales, juste avant le changement suivant.
On constate qu'il y a une dérive causée par le changement de température du chip quand les réglages changent la dissipation.

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 gezwam in de microfoon van een jappenkoopbak met default menu-instellingen en een antenne uit blisterverpakking eraan geknoopt.

Er zijn vier mogelijkheden: 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 grootst. De opgegeven waarden zijn eindwaarden vlak voor de volgende wijziging.
We zien hieraan dat er verloop is ten gevolge van temperatuurwijziging op de chip bij dissipatiewijzigende instellingen.

Le cristal dérive de 40 Hz entre les extrêmes. Cela n'a donc rien à voir avec la température ambiante. Il se pourrait donc que la fréquence diminue en cas de réglage croissant de la fréquence du VFO, quand le réglage change les limites de la plage au lieu d'incrémenter. Afin de réduire cet effet il semble être impératif de se tenir aux recommandations du fabricant, comme mentionnées, à savoir: garder la fréquence du VCO basse et HS_DIV haut quand il y a des choix multiples pour une fréquence de sortie souhaitée.

La conclusion est qu'il est nécessaire d'avoir plus que les 35 réglages des diviseurs afin de garder le VFO du côté bas en évitant les bonds trop larges de haut en bas dans la plage. D'un autre côté, les interruptions de 10 ms sans signal en cas de changement de fréquence >3500 ppm sont un mal nécessaire que l'on voudrait bien éviter, mais diviser la plage du VFO en plusieurs plages ne changera pas le nombre d'interruptions toutes les 7000 ppm sur la plage totale de réglage. C'est une constante.

Je considère la stabilité importante, surtout tenant compte des applications planifiées, d'où le mode opératoire comme décrit dans le paragraphe suivant.

Choix des plages

On peut trier tous les produits HSDIV*N1 possible en ordre croissant. Cela détermine la plage de fréquence correspondant à chaque produit, en effet cela vaut (4.85 à 5,67 GHz)/(HSDIV*N1). Un certain nombre de produit HSDIV*N1 sont égaux entre eux, tandis que les facteurs N1 et HSDIV diffèrent, comme on vient de le voir dans l'exemple de mesure. Quand les produits sont égaux, seul celui avec le plus haut HSDIV est noté, selon la recommandation du fabricant. Ce qui donne 276 produits triés, avec les plages de fréquence chevauchants correspondants.

On ne veut ensuite pas perdre du temps en cherchant en comparant tour à tour quel produit HSDIV*N1 potentiel le plus bas correspond à notre fréquence notée en 4 bytes binaires, mais pour l'optimalisation du temps de recherche, on veut faire cela de manière indexée.

Les plages de fréquence basses (donc avec HSDIV*N1 élevé) se jouxte de près. Toutes les bandes ont effectivement une plage où la fréquence la plus haute égale 5,67/4,85 (les limites du VCO) fois la plus basse. On veut soustraire la déviation de la fréquence nominale de 3500 ppm de chaque côté de la plage calculée, de façon à ce qu'une fréquence nominale choisie peut toujours être déviée dans la limite des 3500 ppm sans transgresser le jeu des limites de plage déterminées par HSDIV*N1. Cela économise un contrôle et donc du temps de calcul. Les limites nominales réglable du VCO deviennent ainsi: 4850*(1+3500E-6)= 4867 MHz et 5670*(1-3500E-6)= 5650 MHz.

A chacun des 267 produits entre 6 et 1408 triés par ordre croissant de HSDIV*N1 correspond maintenant une plage donnée, à savoir: 4867/(HSDIV*N1) à 5650/(HSDIV*N1) MHz. Ensuite il faut choisir un de ces produits correspondant à la fréquence choisie du VFO et ce en le moins de temps possible. Donc de façon indexée. On prend pour cela huit bits des deux premier bytes de la fréquence du VFO notée en binaire. La plus grande plage de l'index est obtenue quand on prend les 8 bits après les premier six bits, à condition que les six premiers soient tous 0. Cela donne une plage de fréquences plu basses. Vérifié par le programme rfreq.c si dans ces circonstances les conditions suivantes soient toujours remplis:

a) Toute plage de fréquence choisie par un produit N1*HSDIV tombe dans une entrée de l'index. On n'a donc jamais une plage de fréquence spécifiée par un index qui ne couvre pas l'entièreté de la plage spécifiée par cet index. Une plage de l'index est alors: 000000[8bitsindex] complété par 18 bits – pour faire un total de 32 bits – allant de tous des 0 à tous des 1. Dans l'exemple: 000000[8bitsindex]000000000000000000 à 000000[8bitsindex] 111111111111111111.
b) Pas besoin de contrôler si les limites du VFO sont dépassées en cas de changement à l'intérieur des 3500 ppm.

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 frequentie-instelling, bij wijziging van een instellingsgrens de frequentie daalt in plaats van stijgt. Om dat effect te minimaliseren lijkt het geboden zich inderdaad te houden aan de aanbevelingen van de fabrikant, zoals genoemd, te weten: 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 beïnvloeden. Dat is een constante.

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

Lage frequentiebanden (dus met grote HSDIV*N1) liggen dicht bij elkaar. Alle banden hebben immers de range dat de hoogste frequentie 5,67/4,85 (de VCO-grenzen) maal de laagste is. De deviatie 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: 4850*(1+3500E-6)= 4867 MHz en 5670*(1-3500E-6)= 5650 MHz.

Bij elk van de 267 tussen 6 en 1408 op grootte gesorteerde producten van HSDIV+N1 is nu de bijbehorende band bekend, 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 geïndexeerd. 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) Pour chaque fréquence la combinaison du HSDIV le plus haut possible avec le N1*HSDIV le plus bas possible est choisie.
d) Chaque fréquence choisie est couverte par cette technique.

Avec 256 nombres d'index choisis de cette façon on ne couvre pas l'entièreté de la plage de fréquence jusque 945 MHz, à savoir les produit N1*HSDIV plus petit que 80. C'est au-dessus de 67,5 MHz. Dans ce cas, les six premiers bits de la fréquence notée en binaire ne sont pas tous 0 non plus.

Il est donc nécessaire de choisir un deuxième index pour le cas où les six premiers bits ne sont pas tous 0. Cet index consiste du troisième jusqu'au dixième bit de la fréquence. On couvre ainsi jusque 945 MHz mais avec quelques complications, à savoir que les index 16, 48, 96 et 192 n'indiquent pas de plage de fréquence couverte par un produit N1*HSDIV. Aussi ces index sont-ils séparés en deux plages par un "exception handler" afin de trouver le produit correspondant. Ce qui nécessite des mesures supplémentaires et une procédure un peu plus compliquée pour les fréquences au-dessus de 65 MHz.

De cette manière on peut trouver, sans devoir deviner, directement la combinaison idéale de N1 et HSDIV.
Cela coute un peu d'espace mémoire pour le tableau, mais il y en a plein dans le contrôleur, et la recherche de la combinaison HSDIV*N1 optimale (la plus basse possible) en devient aussi court que possible.

Travail de calcul pour le contrôleur

Chaque entrée dans le tableau des produits N1*HSDIV contient pour cette plage-là les deux diviseurs HS_DIV et N1, déjà codés dans le format requis par les registres reg8 et reg7 du Si570. Des plages qui se jouxtent peuvent souvent avoir le même pair de diviseurs. En outre chaque entrée devrait contenir un nombre en 4 bytes qui est le produit des deux diviseurs divisé par la fréquence du cristal. Ce dernier nombre doit à chaque changement de l'indication du LCD être multiplié par la représentation binaire de cette indication, qu'on a calculée au préalable à l'aide du tableau multitab et qui est calculé immédiatement à chaque changement de l'indication.

Il s'avère que la fréquence du cristal de chaque exemplaire du Si570 est légèrement différente, aussi n'est-il pas possible de programmer un tableau fixe dans un microcontrôleur qui doit fonctionner avec n'importe quel Si570. La valeur RFREQ requise pour le facteur d'échelle doit effectivement être calculée selon la formule de la **figure 1**:

RFREQ= LCDfreq * {N1 * HS_DIV/Xtal}

Le diviseur total N1 * HS_DIV entre 6 et 1408 pour toutes les versions du Si570, a été encodé dans le tableau. Le RFREQ doit alors être multiplié par le diviseur total N1*HS_DIV et divisé par la fréquence du cristal comme mesurée par la calibration.

Le processus final est comme suit:

1. Déterminer la valeur binaire en 32 bits [0,31] de la fréquence indiquée au LCD. Si la fréquence franchit une limite de plage (le choix de la bande est expliqué plus loin) alors l'indication est remise à la limite de la plage avant de coder la fréquence en BCD.
2. Déterminer à partir de la valeur binaire de la fréquence un index en 8 bits en prenant la partie comme discuté plus haut.
3. C'est l'index dans un tableau de moins de 255 entrées, et ce tableau donne l'adresse dans un autre tableau prodtab des produits. Ce dernier tableau contient 4 bytes par entrée, soit 2 bytes pour N1*HS_DIV et deux bytes contenant N1 et HS_DIV codés comme prescrit par le fabricant du Si570.
4. La fréquence binaire est d'abord multipliée par N1.HS_DIV de l'entrée du tableau. Cela donne une valeur qui ne varie pas beaucoup parce que quand la fréquence augmente, HS_DIV*N1 diminue.
5. La valeur de 1/Xtal du Si570 utilisé est connue (par une étape de calibration à discuter). Cette valeur est multipliée par le résultat de l'étape 4.

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 frequentierange 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 exception 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 geïndexeerd 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 vermenigvuldigd 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 **figuur 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 kalibratie vastgestelde kristalfrequentie. 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 varieert, omdat bij toenemende frequentie HS_DIV*N1 daalt.
5. De waarde van 1/Xtal voor de gebruikte Si570 is bekend (uit een te bespreken kalibratiestap). Die waarde wordt vermenigvuldigd met het resultaat van stap 4.

6. N1 et HS_DIV sont collé à ce résultat selon la codification du Si570 du tableau prodtab.
7. Le tout (6 bytes) est envoyé via I2C au Si570 pour programmer la fréquence.
8. La différence de fréquence par rapport à la dernière fréquence centrale est retenue. Quand la limite des 3500 ppm est franchie, alors la fréquence centrale est à nouveau établie.

Multiplier

Dans ce cas, on a besoin d'un multipliant de 4 bytes de largeur. Multiplier deux nombres de 4 bytes produit un résultat de 8 bytes. Ce processus peut être limité quand des bytes sont 0. Et comme on l'a appris à l'école primaire ainsi fonctionne la multiplication de grands nombres ici. On faisait cela avec les tables de multiplication jusque 10. Ici on a une instruction de multiplication de 8 bits de large dans le contrôleur, ce qui nous permet de multiplier en utilisant les tables de multiplication jusque 256. Pour deux nombres 32 bits on a besoin d'au maximum 16 multiplications avec les additions correspondantes. C'est dans le programme et afin d'en limiter la taille, l'adressage est indirecte. Cela requière bien 4 pointeurs d'adresse, tandis ce que le contrôleur n'en a que deux. Mais ce contrôleur dispose de quatre bancs de pointeurs. En échangeant des bancs on obtient ainsi deux jeux de 2 pointeurs.

Tester et déboguer, jusqu'à ce que quelques multiplications fonctionnent correctement s'avère nécessaire.

Etapes suivantes dans le développement

Une routine de division est toujours nécessaire, à savoir pour calibrer l'ensemble; il faut calculer 1/Xtal. Diviser un nombre de 4 bytes par un diviseur de 4 bytes marche également comme appris à l'école primaire, si ce n'est qu'ici c'est implémenté bit par bit. Que les gosses de l'école primaire ne savait pas qu'on ne compte pas par dix tables de dix entrées, mais qu'il suffit de connaître seulement deux tables de deux entrées, à savoir la table de 0 et la table de 1, quand on calcul en binaire, n'est que juste, sinon on aurait eu des manifestations des enfants concernant la qualité de l'enseignement. Le système décimal fait avancer les choses, mais que penser du système 256 pour lequel il y a également une instruction de division dans le contrôleur? Laissons de côté, je veux construire un VFO utilisable et non devenir un mathématicien. En outre la phase de calibration n'est à faire qu'une fois et n'a pas besoin d'être super rapide. Le quotient comporte 5 bytes ce qui fait amplement de chiffres significatives pour calculer le facteur 1/Xtal.

Suite des tests et tricotage

Je me demande à quelle vitesse le changement entre deux fréquences de plage différentes et donc en dehors de la plage des 3500 ppm range s'opère. Cela peut être examiné en remplaçant le "changed bit" sur lequel normalement la routine principale attend, et qui indique que l'actuateur a été actionné, par une inversion de tension sur une broche de sortie, de laquelle la fréquence peut alors être mesurée. Cela montre que tous le charabia y inclus l'affichage de la fréquence et l'affichage diagnostique des 16 caractères hex RFREQ, se fait 260 fois par seconde. Même si on tourne le bouton davantage plus vite (plus que 2,5 tours par seconde), cela ne cause pas de problème, car la fréquence est mise à jour correctement à chaque interruption par le bouton en mémoire vive en haute priorité, de façon à ce que la boucle suivante de calcul utilise la dernière valeur de la fréquence. Il n'y a donc pas de dérive en tournant le bouton trop vite causée par le calcul et le pilotage du Si570.

Ces calculs coutent maintenant moins de 4 ms tandis ce que le Si570 lui-même a besoin de 10 ms pour effectuer un changement dans ce cas.

Comme on est occupé à mesurer, voyons combien de temps ça coute de remplir le LCD de deux ligne de 16 caractères chaque (diagnostic routine test_LCD a été écrit pour cela). Cela se fait 452 fois par seconde, donc à peu près 2 ms pour un affichage complet et 1 ms pour la ligne complète de la fréquence de 16 positions. Avec un encoder optique à 100 pas par tour, il faudrait alors tourner le bouton plus de 5 fois 360°

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 kalibreren 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 geïmplementeerd. 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 zesjarigen 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 kalibratiestap eenmalig en die hoeft helemaal niet supersnel te zijn. Het quotiënt 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 dezelfde 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 ten gevolge 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

par seconde pour que l'affichage ne puisse plus suivre sans dérive. Mes yeux sont beaucoup plus lents, mais on ne peut pas les prendre comme référence après avoir observé le fond d'un verre de Bols pour rendre la soudure des SMD possible.

Tous les tableaux ont été programmés en C, dans un programme qui produit ces tableaux en listing assembler. On ne peut effectivement se permettre des erreurs dans ces tableaux, cela causerait une fréquence du VFO différente de ce qui est affiché. De surcroit, cela nous économise beaucoup de dactylographie.

Détails de la calibration

Regardons de plus près la calibration. L'intention est de régler la sortie du VFO "zero beat" avec un signal d'étalonnage de 10 MHz, WWV ou un compteur de fréquence bien étalonné, peu importe. Le VFO se met, quand on le branche en appuyant sur le bouton de commande, en état de calibration. C'est indiqué par la deuxième ligne de l'affichage. Il le fait par ailleurs également si la somme de vérification des données en EEPROM serait erronée au moment qu'on branche le VFO. C'est par exemple le cas quand on le branche pour la toute première fois après l'avoir construit ou encore une ou deux fois par ans pour des raisons pas claires. Une faille souvent rencontrée par les EEPROM's de contrôleurs. **Figure 6** présente une explication.

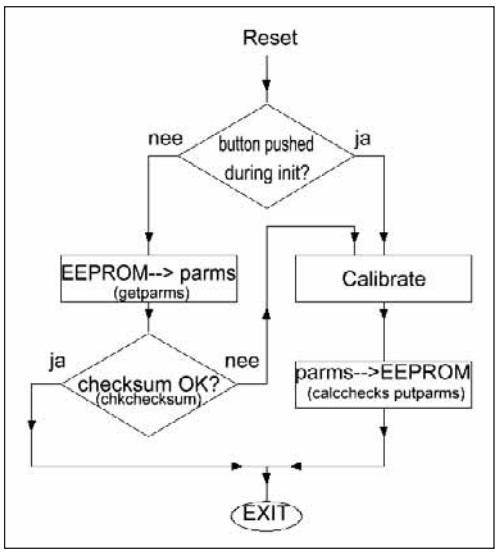


Fig. 6. EEPROM et/en checksum

On peut régler la fréquence, indiquée au LCD à exactement 10 MHz, pendant que le VFO tourne aux alentours de 10 MHz, jusqu'à ce que sort exactement 10 MHz de la sortie. Le battement est de maximalement 20 kHz au-dessus et au-dessous, parce que le fabricant spécifie cela comme tolérance sur le cristal. Qu'on tourne vite ou lentement ne change guère à la calibration, c'est toujours 10 Hz ou 1 Hz par pas. L'affichage indique, quand la sortie (après une période de chauffe) est "zero beat" avec le signal d'étalonnage, alors une autre valeur, proche de 10 MHz, car les cristaux des Si570 diffèrent. La limite de plus ou moins 20 kHz, prévient aussi que les diviseurs N1 et HS_DIV ne correspondent plus à 10 MHz. Quand la sortie est exactement "zero beat" avec le signal d'étalonnage de 10 MHz, on pousse à nouveau sur le bouton de commande. La valeur 1/Xtal pour le Si570 utilisé est alors calculée selon la formule:

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

N1*HS_DIV est maintenant connu. Comme 10 MHz en hex est 00986080 et que l'index utilisé dans les tableaux est donc 38, alors à 10 MHz ces diviseurs sont 6*82=492. Le RFREQ dans le numérateur est déterminé en lisant le Si570, pendant qu'il produit 10 MHz. Ces deux nombres sont alors divisés par la routine 4byte/4byte, ce qui détermine le quotient 1/Xtal. Ce nombre est alors stocké dans la mémoire tampon de 96 bytes RAM et peut être utilisé directement par le VFO. De plus ce tampon RAM est également écrit dans le EEPROM avec la somme de vérification,de manière à permettre au moment qu'on branche le VFO à nouveau quand les paramètres sont chargés à partir du EEPROM un contrôle pour vérifier qu'il n'y aient pas de falsification du contenu. Si ce serait le cas, le VFO se met automatiquement en mode calibration. Après chargement à partir de l'EEPROM du tampon de paramètres et éventuellement une nouvelle calibration, le VFO se met en mode normal.

Cette routine n'est nécessaire que pour la calibration et alors il n'y a pas de multiplications, ce qui permet d'utiliser les mêmes bytes RAM que pour la multiplication. "Us bin sunig", mais ça, les belges le savaient déjà.

(à suivre)

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 intypen en uitzoekwerk uit.

Details kalibratie

Nu eerst eens nader kijken naar de kalibratie. De bedoeling is de VFO met zijn output zero beat te zetten met een 10 MHz ijksignaal, WWV of een goede gekalibreerde teller, maakt niet uit. De VFO komt, als je tijdens inschakelen van de netspanning de bedieningsknop al ingedrukt hebt, in de kalibratiestand. 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. **Figuur 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 kalibratie 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, volgt daaruit 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 twee getallen worden dan gedeeld in de 4byte/4byte deelroutine, waarmede als quotiënt 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 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 kalibratiemode. Na laden uit de EEPROM van de parameterbuffer en eventueel benodigde kalibratie, gaat de VFO over in de normale mode.

Die deelroutine is alleen maar nodig voor de kalibratie 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.

(wordt vervolgd)