Een dual Si570 VFO (deel 3, slot)
Un double VFO Si570 (3ème partie, fin)

Door/par PA0WV – Traduit par/vertaling ON3PX

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 kopiëren 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 breedbandversie 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 even lang. 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 checksum die in het eerste byte wordt gezet is de som zonder carry van alle overige bytes van de buffer (of, anders geformuleerd: de rest van de deling van de som van die bytes door 256), met een minteken ervoor gezet. 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.

Limites de bandes

Chaque modification de la fréquence affichée se produit sous interrupt. Le programme principal pourrait justement être en cours de modification de la fréquence. Pour empêcher cela, la fréquence affichée est copiée dans un buffer de 9 bytes, avant de les traiter et pendant la recopie la routine d'interruption de l'actuateur est suspendue durant un court instant.

A chaque calcul de la fréquence binaire on vérifie avant si la fréquence BCD de l'afficheur ne tombe pas en dehors des limites de bande. Pour la version CMOS, elles sont forcement 3445khz et 160Mhz, mais pour les fréquences de gammes amateur on reste dans ces gammes quand on choisit ces bandes au moyen de la roue codeuse optionnelle qui agit sur les 4 entrées P0.0 à P0.3 du contrôleur. Cette roue codeuse choisit de 0 à 9, donc maximum 10 bandes. Si vous utilisez des ponts ou une carte à interrupteurs tumbler avec les valeurs 1, 2, 4 et 8, ou un codeur décimal qui parcourt de 0 à F, vous pouvez obtenir jusqu'16 bandes. En augmentant la fréquence au dessus de la limite supérieure, il y a refus, même réaction si l'on tente de passer sous la limite inférieure. Les limites pour 16 bandes sont enregistrées en valeurs BCD de 9 bytes dans le programme. 13 bandes de 80 mètres jusqu'à 70 cm, ainsi qu'une version large bande de 3445 khz jusqu'à ce que le CI permet (chez moi donc 160Mhz.) Si le sélecteur de bandes, lors de la calibration, ne se trouvait pas sur une bande qui comprend 10 Mhz, il ne faut pas s'en préoccuper. Indépendamment de la gamme choisie, le LCD se place sur 10 Mhz.

La EEPROM

Le contrôleur contient une EEPROM. On peut programmer par byte, ce qui prend 4 ms par byte, ou par page de 32 bytes, ce qui prend le même temps. On choisit donc ce mode; la lecture est beaucoup plus rapide et ne se peut que par byte.

La EEPROM est plus ou moins protégée contre l'écriture, et malgré que j'utilise la protection de manière optimale, selon les prescriptions du constructeur, il m'arrive avec d'autre projets, que j'utilise quotidiennement qui l'utilisent (p.ex. Kujer2), que la EEPROM ne fournit pas les données entrées. Le plus souvent après un bel orage. Je constate cela car une checksum est programmée dans le contenu et elle ne correspond pas lors de la lecture. D'où en sauvant le buffer de paramètres de la RAM, j'en calcule la checksum, et je la programme ensemble avec avec les données du buffer dans l'espace des pages entières (multiples de 32 bytes) dans la EEPROM. La checksum que l'on place dans le premier byte est la somme sans carry de tous les autres bytes du buffer (autrement dit : le reste de la division de la somme de tous ces bytes par 256), précédé du signe moins. Lors de la lecture de la EEPROM la checksum calculée de tput le buffer RAM incluant la checksum chargée doit être. Cela semble un peu compliqué, mais j'ai fait cette inversion pour empêcher que tous les contenus par défaut, comme tous les bytes 00, ne fourniraient une checksum correcte.

Cette question est testée en écrivant les nombres 1 à 63 dans le buffer de la RAM, d'en calculer la checksum et de la placer dans le byte 0, de programmer ensuite en EEPROM, d'effacer le buffer RAM, de lire le buffer dans la EEPROM, et d'écrire les 8 premiers bytes sur l'afficheur LCD. Nous avons calculé nous-même la checksum au moyen de la formule retenue de la ULO pour la somme d'une série arithmétique, et elle correspond (après avoir éliminé un bug).

Il est donc important de vérifier soi-même chaque stade de développement. En fait, cela ne diffère pas de construction d'un kit, où à chaque pas il convient de vérifier chaque pas. Bonne valeur de résistance, électrochimique à l'endroit, controle des soudures. A chaque niveau du contrôle de soi-même est extrêmement important.

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 aangeven, 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, dan 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 ampère 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 à 500 mA. In paniek de stroom verbroken, en de Si570 heeft het blijkbaar overleefd, voor zover ik dat kan vaststellen.

Mémoire de fréquence

Il est fastidieux à chaque mise sous tension de rechercher chaque fois une fréquence de départ. C'est pour quoi une mémoire de fréquence est incluse. Si pendant le fonctionnement du VFO on appuie sur le bouton, la fréquence de l'afficher LCD se retrouve dans la EEPROM. Plus tard, en démarrant le VFO sur cette bande, cette fréquence est immédiatement reprise. Si pour cette bande aucune fréquence n'a été sauvée, il reprend la fréquence inférieure de cette bande comme fréquence de départ, car c'est le début de la bande CW. En tout, il y a 16 fréquences possibles, une par bande. Si vous changez de bande, vous retrouvez directement la fréquence présélectionnée. S'il n'y en a pas, vous obtenez la fréquence inférieure de la bande choisie.

Dans le buffer de la RAM du contrôleur exprimée en pages autorisées de la EEPROM, est limitée à 3 pages, soit 96 bytes alloués pour ce but. On y retrouve déjà la checksum, les 4 bytes de 1/Xtal et un offset de moyenne fréquence, il en reste donc 87. Cela ne suffit que pour y mettre9 fréquences de 9 bytes et pas 19. C'est pourquoi les fréquences sont sauvez en format packed BCD de 5 bytes, qui après lecture sont décodées en format BCD normal. Ce qui procure une réserve de 7 bytes dans le buffer de paramètres RAM.

Les limites 3500 ppm

Chaque fois que l'on règle une nouvelle fréquence, on examine si cela est possible "au vol", dans les cas où la fréquence centrale est moins de 3500 ppm. Lors de l'installation centrale on en retient quelques bytes pfn dans la mémoire. A chaque RFREQ recalculé on voit si celle-ci ne dévie pas plus de 0x0.26 ce qui en gros, dans le pire des cas représente 3500 ppm. Si oui, RFREQ peut être modifié au vol, si ce n'est pas le cas la nouvelle fréquence est choisie comme fréquence centrale et les bytes pfn, qui donnent la dernière fréquence centrale, sont mis à jour avec la nouvelle valeur.

SDR

Si l'on expérimente avec la software defined radio un VFO qui donne deux signaux déphasés de 90 degrés est indispensable. Également pratique pour les émetteurs SSB système phasing. Cela se fait normalement avec un double D flip-flop en compteur Johnson. Le VFO doit fournir une fréquence quadruple de celle affichée à l'écran. En corollaire, cela implique que cette lecture ne peut pas indiquer plus de 40 MHz, si le VFO va jusqu'à 160 MHz.

MF

Dans les cas de la construction d'un transceiver, la fréquence de réception est celle de l'oscillateur moins la moyenne fréquence. D'où; en P3.6 une entrée commutable. A l'état haut, l'émetteur est actif et à l'état bas l'on ajoute la valeur de la moyenne fréquence pour utiliser le VFO comme oscillateur local alimentant l'étage mélangeur.

On obtient donc une fréquence d'oscillateur réelle (parfois beaucoup) plus haute que celle que l'afficheur fournit. Pour cette raison on contrôle si cette valeur ne dépasse pas la valeur maximale admise du Si570 (chez moi 160 MHz). Aussi non seulement le Si570 n'est pas commandé, mais un message overflow remplace l'indication de fréquence de réception.

Prudence est mère du magasin de porcelaine

En lisant cette histoire, il est clair que je travaille prudemment, pour ne pas abîmer des composants et je me tiens donc aux spécifications d'usine. Celle-ci n'a pas avantage à fournir des specs meilleures que réelles. Vous êtes donc certains que l'ensemble est de construction reproductible.

Il y a pourtant ce cas : un ampèremètre analogique d'un demi Ampère fond d'échelle indique une consommation du Si570 de 100 mA. Je donne un coup sec à l'actuateur et le courant semble parfois sauter vers 300 à 500 mA. Paniqué, j'ai coupé le courant, et le Si570 semble avoir survécu à l'aventure, selon mes constatations.

Als er wat fout gaat in de software, zoals RFREQ buiten zijn grenzen, of een ack ontbreekt van een van de 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

Het duurt ongeveer 2 ms om de display 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 displaytijd 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 brandstofbesparing onze hobby-uitgaven te kunnen betalen. Verversen 25 keer per seconde zou genoeg zijn, dus daar is op twee fronten tijdwinst 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 er om de milliseconde een karakter uitgehaald en in de display gezet. De karakterspatie in de tijd is dan zo groot dat niet op de busyvlag 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 interruptafhandeling gebruikt om de cursor van de display op de juiste lijn te zetten. Is er geen wijziging in de displaylijn-RAMbuffers, dan gebeurt er niets.
Wordt tijdens de display van een lijn, de lijn inmiddels ververst in het displaygeheugen 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.

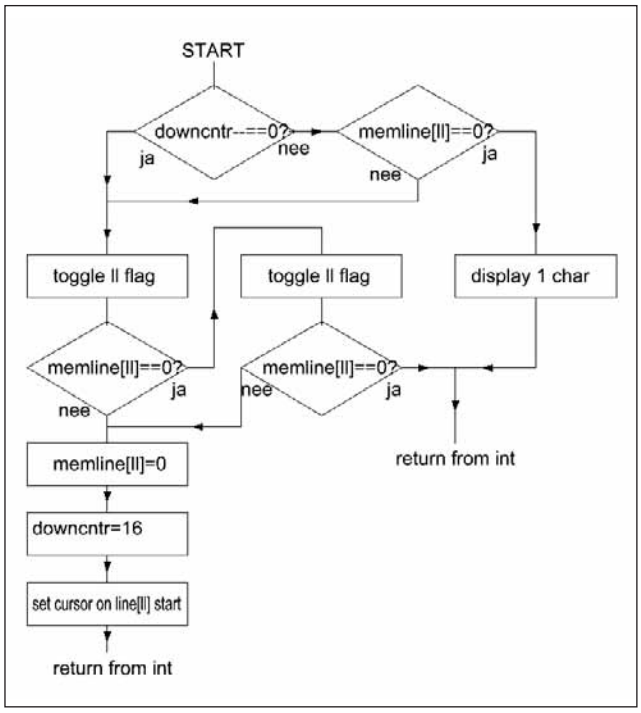


Fig. 7. Flow chart van de display interrupt afhandeling
Fig. 7. Flow chart de la routine d'interrupt d'affichage

In **figuur 7** wordt de werkwijze van de interruptroutine nader toegelicht. Uitgewerkt plan, echter niet geïmplementeerd.

Si un défaut apparaît dans le software, comme RFREQ hors limites, ou si un ack des routines de commandes I2C manque, cela est annoncé par un message d'erreur sur la ligne inférieure du LCD. Ces rares phénomènes semblent se produire parfois quand le Si570 reçoit une nouvelle demande de fréquence éloignée via I2C, pendant ce temps-là il ne fournit pas de signal, selon le constructeur moins de 10 ms. Pendant ce temps le programme continue à tourner, et si celui-ci demande une nouveau réglage de fréquence via le I2C, c'est la foire, du moins de temps à autre. C'est une mauvaise conception, du moins de Silicon Labs, car le bus I2C peut tenir l'horloge à l'état bas au travers de l'esclave (clock stretching) quand il n'est pas prêt, et cela ne se produit apparemment pas, en conclusion provisoire.

J'ai résolu l'affaire en démarrant, dans les cas de demande de nouvelle fréquence centrale du Si570, un décompteur de 10 ms qui ramène l'interrupt timer_0 à zéro au bout de 10 ms. Si le programme principal intervient entretemps pour une nouvelle fréquence centrale au bus I2C avant que ces 10 ms sont passées, alors on attend simplement qu'elles soient passées, à condition que l'actuateur n'a pas été déplacé. Si c'est malgré tout les cas, alors le Si570 n'est pas règlé, mais on calcule la nouvelle valeur qui est présentée dans des conditions identiques au bus I2C.

L'afficheur

Cela prend environ 2 ms pour remplir complètement l'affichage. C'est un temps équivalent pour le recalcul complet de la valeur suivante du Si570. Un partie du temps d'affichage est de la perte d'attente, car après le chargement d'un caractère il attend qu'un flag 'busy' se libère. D'autre part, en tournant sans arrêt au bouton actuateur, la boucle principale est parcourue une centaine de fois par seconde et l'affichage s'en trouve continuellement rafraîchie. Comme nous ne sommes pas capable de lire à ce rythme, que les cristaux du LCD ne battent pas des records de vitesse, surtout dans notre habitation soumise à des restrictions de chauffage pour pouvoir assumer nos dépenses pour notre hobby. Un rafraîchissement de 25 fois par seconde suffirait., il y a donc du temps à gagner sur deux fronts. Nécessaire ou non, allons-y.

C'est faisable en écrivant la sortie vers l'afficheur dans la mémoire et non dans l'afficheur. Sur base d'un interrupt (timer_1 overflow) qui reprend un caractère chaque milliseconde et le place sur l'afficheur. L'espacement des caractères dans le temps est alors si grand qu'il ne faut plus attendre le flag 'busy'. Chaque ligne est traitée à son tour, il est donc impossible que si on rafraîchit sans cesse la première ligne, la seconde n'ait pas son tour. Une ligne qui n'a pas été modifiée n'est pas représentée à l'afficheur. Si une ligne est terminée au bout de 16 interrupts, le compteur de curseur est replacé sur le coté gauche de l'afficheur et une routine de fin d'interrupt est utilisée pour placer le curseur de display sur la ligne correcte. S'il n'y a pas de modification dans les buffers RAM d'affichage, il ne se passe rien.
Si pendant le temps d'affichage d'une ligne cette ligne est rafraîchie dans la mémoire de l'afficheur à cause du parcours de calcul, l'interrupt suivant repart au début de la ligne. Tout ceciest rendu possible au moyen de quelques flags qui reflètent l'état de fonctionnement, et à cause de la routine principal qui place un flag memline[2], qui à la fin de l'interrupt suivant de la ligne d'affichage adéquate est remis à zéro.

La **figure 7** illustre plus clairement la façon de travailler de la routine d'interrupt. Un plan plus détaillé n'est pas fourni.

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) en 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 (**figuur 8**). Zo leek het 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 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 inputpen 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

Offset CW et MF

Si vous voulez utiliser ce VFO dans un transceiver, vous ne pouvez pas émettre et recevoir avec la même fréquence VFO, parceque là vous seriez au battement nul. Pour cette raison une broche est réservée, haute pour l'émission (option défaut) et en réception au niveau bas. L'intention, c'est d'englober l'offset, aussi bien de la moyenne fréquence (0 chez les récepteurs DC) que la différence avec le battement nul, positionné avec précision au centre de votre bande passante de votre filtre. Pour cette raison cet décalage est règlable et s'ajuste lors du stade de calibration.

Deux VFO A et B

On peut étendre les possibilités (**fig. 8**). Cela me semblait attirant d'utiliser 2 VFO commutables de l'un à l'autre au passage d'émission et de réception. Dans ce cas vous pouvez, en immobilisant stablement votre fréquence de réception, faire varier indépendamment celle d'émission de 1 kHz ou tout près, question de faire plaisir aux OMs lors de pile-up. En fait cela offre 4 possibilités :

- émettre avec A, recevoir avec A
- émettre avec A, recevoir avec B
- émettre avec B, recevoir avec A
- émettre avec B, recevoir avec B

Pour cela on se sert du second buffer de fréquence de 4 bytes, qui peuvent alors être échangés avec le premier en commandant une broche d'entrée. Mais l'affichage de ce second buffer est en BCD 9 bytes, duquel on l'avait déjà calculé. Cet affichage doit donc (à cause du criant besoin de RAM causé par ce genre d'extension) être recalculé à,partir de cette valeur de fréquence binaire. Cela demande un beau calcul, c'est-à-dire diviser le nombre de 4 bytes par 10, le reste c'est le premier digit BCD, encore diviser le quotient par 10, le reste étant le second digit BCD, et diviser neuf fois par 10 pour obtenir les 9 digits.

Auparavant j'ai écrit que la routine de division se faisait de façon binaire, bit à bit, car en n'en avait besoin qu'une seule fois pour la calibration, et que j'en trouvais la programmation plus simple. Mais ce plan téméraire m'a imposé d'utiliser l'instruction machin divAB, qui divise le byte dans l'accumulateur A par B et qui place le quotient dans A et le reste en B. Ce n'est pas tout lisse car après le calcul du reste, car il faut ramener la valeur suivante, comme des morceaux de queue, et cela n'est pas un byte car le reste est déjà un nibble (4 bits), on ne peut donc reprendre qu'un nibble à mettre ensemble avec le reste dans l'accumulateur A. Bref, optimaliser. On a donc encore besoin de 8 instructions divAB par division décimale, donc en tout l'instruction divAB est exécutée 72 fois lors de la transformation inverse vers le BCD 9 chiffres.

On se demande alors s'il y a gain de temps par rapport à la division binaire et si oui, de combien. C'est ce que j'ai examiné. La transformation inverse de 32 bits vers 9 BCD avec la routine écrite à cet effet prend 680 microsecondes. Si on fait cela avec le diviseur binaire bit à bit la division par 10 prend 1,5 ms; comme il faut 9 divisions elles font 13,6 ms. À ajouter le temps de charger et décharger. Le gain de temps en utilisant l'instruction divAB est donc substantiel.

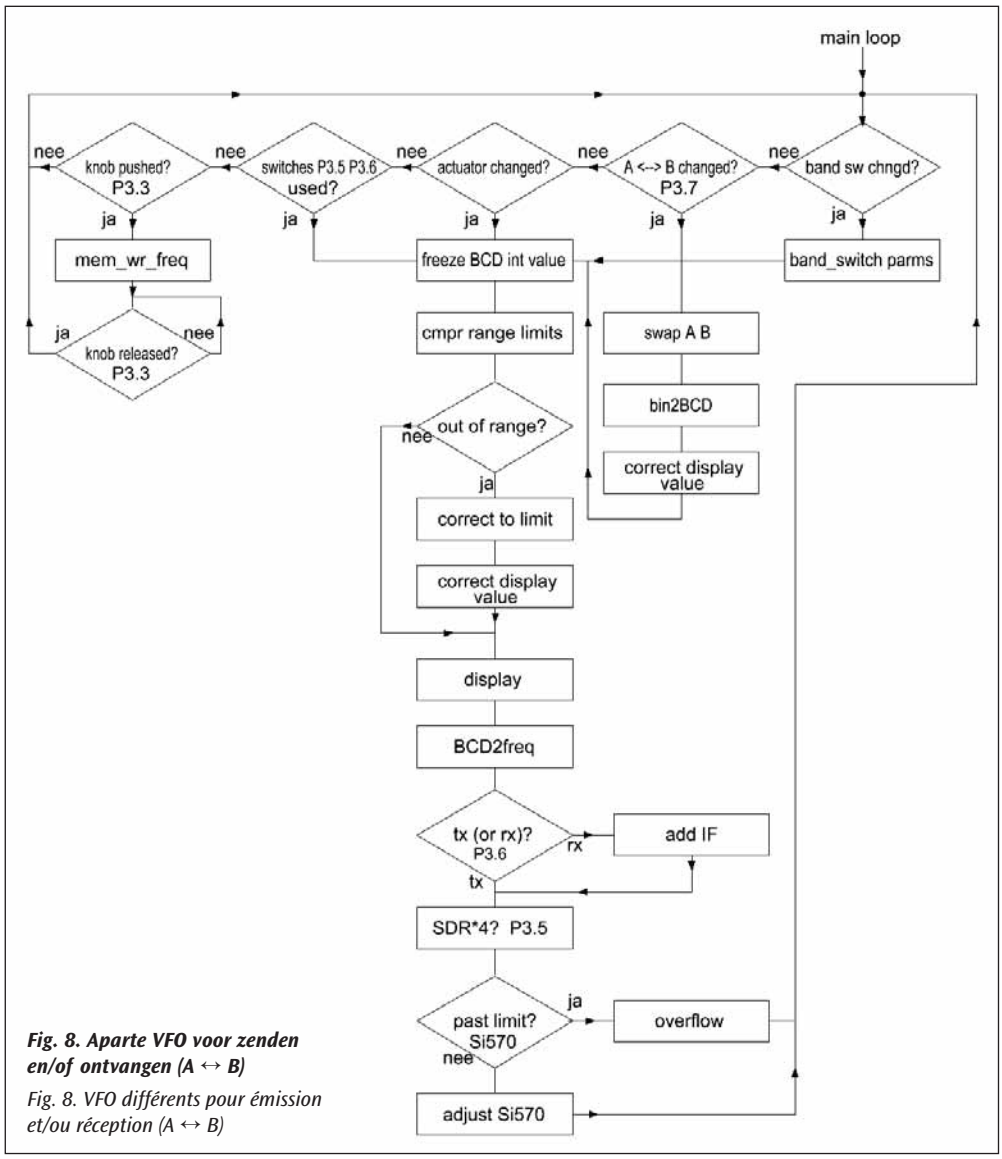


Fig. 8. Aparte VFO voor zenden en/of ontvangen (A ↔ B)

Fig. 8. VFO différents pour émission et/ou réception (A ↔ B)

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 quotiënt 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 (quotiënt) 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, doch 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 ten opzichte van 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.

Implementatie VFO-A en VFO-B

Over de wijze waarop ik dat het best kon implementeren, heb ik overlegd 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 geïmplementeerd.

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 hoofdprogramma 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 main loop 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

Implémentation VFO-A et VFO-B

De la façon de laquelle je pouvais faire la meilleure implémentation, je me suis mis d’accord avec Steef PA3S, une station de contest bien connue, Henk, PA1A également connu, Freud pourrait nous expliquer pourquoi, mais son mât conception perso a récemment été renversé par le vent, bien sur c’est possible, mais il critiquait les plans d’autre concepteurs de mats sur zendamateurs.org, dans ce cas il vaut mieux ne pas donner à mère nature l’occasion de donner des leçons; je le laisserai donc en paix (hi!). Steef PA3S a écrit qu’il ne travaillait pas en split, et sa réponse à mon questionnement était très utilisable et détaillée.

J’ai donc décidé d’ajouter deux points de raccordement, à nommer émettre/recevoir et VFO A ou B. Cela permet toutes les options. A la rigueur, moyennant quelques portes NOR ou NAND, question de ne pas avoir appris les portes digitales pour rien avant de passer l’examen F(ull) “examen”. C’est une application du savoir acquis.

Mesures

La mesure de la durée d’exécution de la procédure d’interruption de l’actuateur s’est faite en couplant la broche interrupt 0 à une broche que le programme principal met à l’état bas puis à l’état haut, de façon à ce que le programme principal ne fait plus que créer des interrupts. La routine de traitement d’interrupt inverse continuellement une broche de sortie, de cette façon un compteur placé sur cette broche mesure le fréquence des traitements d’interrupts quand le processeur les traite à 100% du temps. Ils semblaient se produire 22500 fois par seconde.

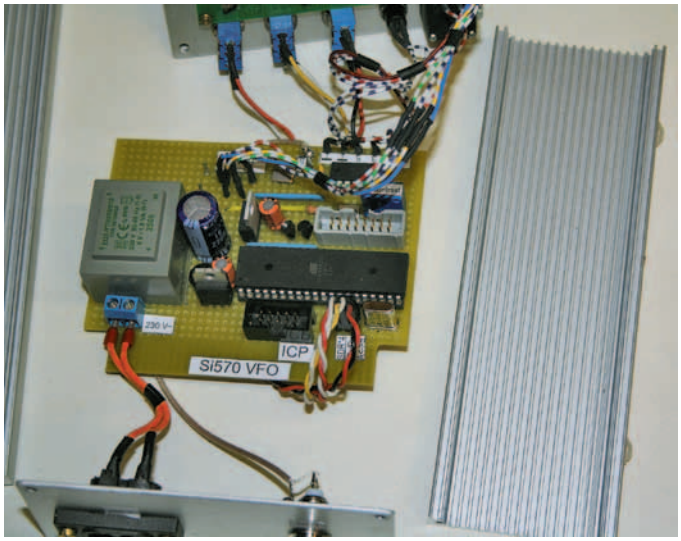
Mesure de la durée d’une correction et de la boucle de réglage dans le programme principal. Pour mesurer cela, le programme n’attend pas le flag ‘changed’ qui est activé lors du traitement d’interrupt, mais celui-là se retrouve continuellement commuté comme si on changeait tout le temps la fréquence. L’affichage est mis à jour et toute la boucle principale est parcourue, la mise à jour du Si570 y compris. Lors de la mesure, cela se produit 340 fois par seconde.

L’ensemble posé simplement sur table, elle se parcourt selon le compteur de pointe à pointe 1,5 Hz à 3445 kHz. Il vaut mieux ne pas toucher le CI Si570, car cela dépend de la température, qui est bien constante dans le schack, aussi longtemps que nous pourrons encore payer le gaz de chauffage, et que le pays inondé de Grunninghe ne se retrouve pas sur la carte (hi !). A la mise sous tension tout se déroule assez bien, mais il faut bien un quart d’heure pour stabiliser l’ensemble. Ne calibrez donc pas avant d’avoir attendu ce temps de mise en service.

Montage

Pour éviter d’écraser une construction personnelle posée à terre (pour éviter qu’elle ne tombe), je l’ai placée dans un boîtier – 6 de chez

Foto 4 / Photo 4



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 **figuur 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 met een grotere boor. Figuurzaagsneden uitvijlen met een platte bastaard tot op de kraslijn tekenrand, daarna afschuren met schuurpapier op een verfroerhoutje.

De geprogrammeerde processor kost 16 euro inclusief verpakking en porto. Neem contact op met mijncall@amsat.org, waarbij mijncall uiteraard vervangen moet worden door

PA0WV

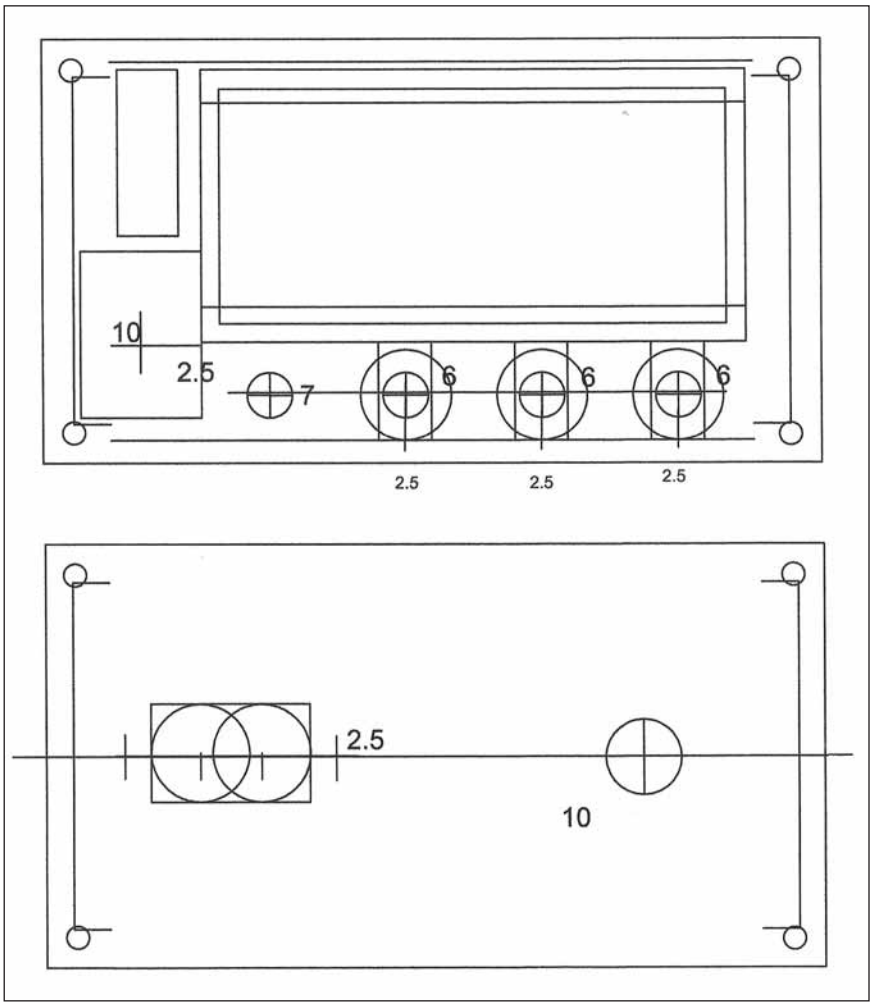


Fig. 9. Boorplan - Plan de perçage

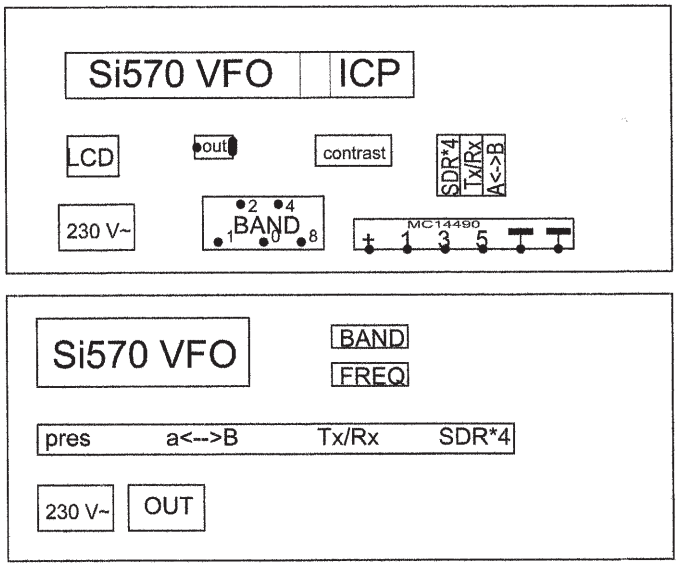


Fig. 10. Etiketten- Etiquettes

Conrad, numéro de commande 523232. Voyez également chez Reichelt, leurs prix sont inférieurs d'environ la moitié. Avec unee imprimante, rapporter les points de perçage de la **figure 9** sur un endroit adéquat sur la face avant ou arrière. Découper, scotcher sur le boîtier. Pointer avec un pointeau, percer avec 2,5 mm, utiliser uniquement de l'alcool comme lubrifiant, ne pas boire. Ensuite repercer au bon diamètre plat, découper les ouvertures carrées à la scie à chantourner. Débavurer les trous avec une mèche de plus grand diamètre. Rectifier les trous découpés au moyen d'une lime bâtarde jusqu'au bord des lignes de découpe, puis peaufiner au papier émeri fixé sur un bâtonnet.

Le processeur programmé coûte 16 euros, emballage et port compris. Contactez-moi sur mycall@amsat.org, en remplaçant mycall par

PA0WV