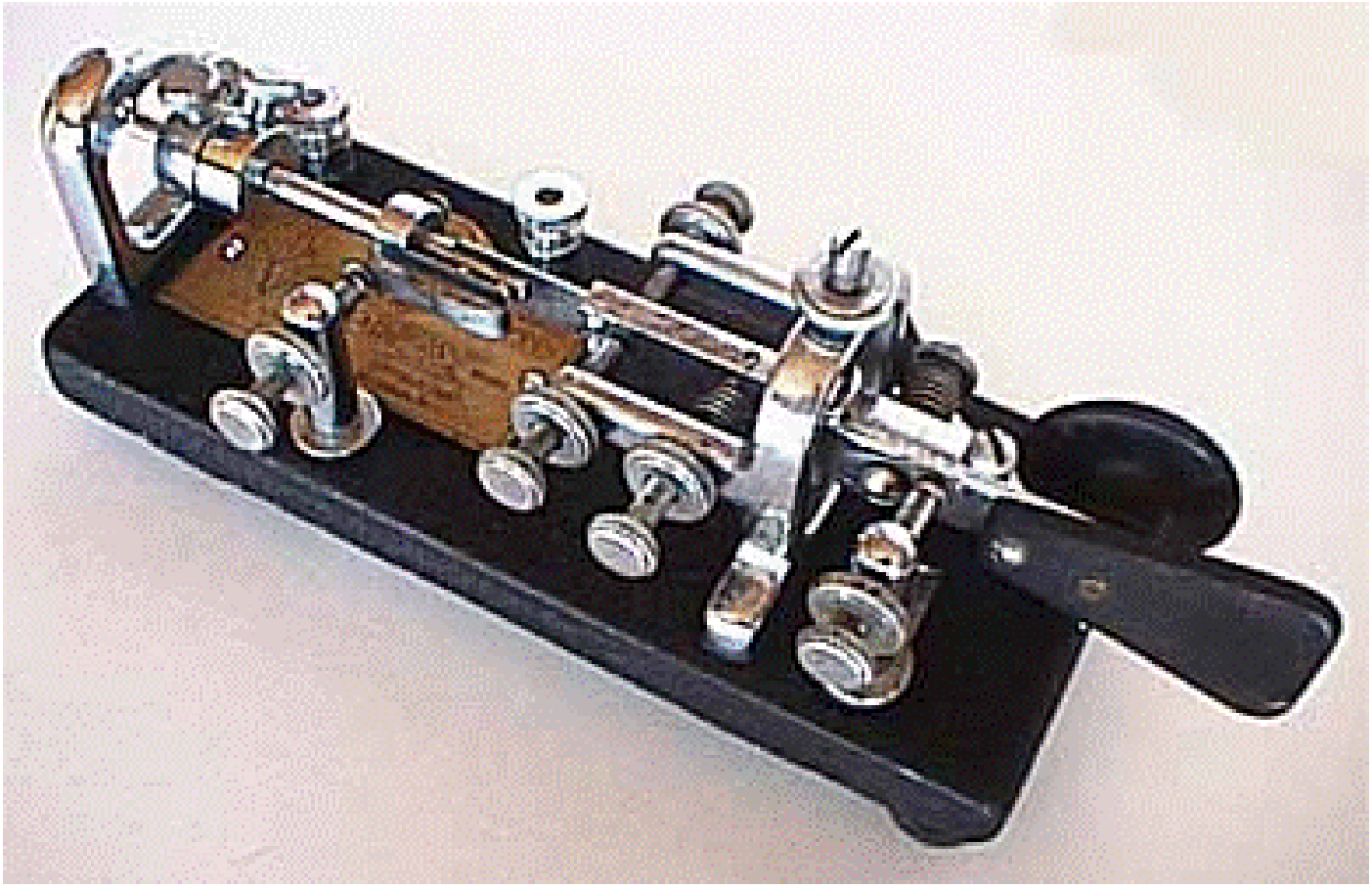
De Bug Butcher

PA0WV

Inleiding

Een bug (vibroplex) geeft door een trillende pendel bij het naar rechts duwen met de duim van de rechterhand een serie punten. Duw je met je wijsvinger naar links dan krijg je een streep zolang je duwt. Er zijn ook bugs voor linkshandigen, dan gaat dat net andersom, omdat links je duim rechts zit.

Het kan aardig zijn met wat elektronica die duw



niet per streep te hoeven maken. Een cijfer 0, vaak voorkomend als je een pa0 call hebt, vereist immers 5 duwacties. Als je een langere duw in strepen kunt verdelen, zoals de bug dat zelf met punten doet, en die dan keurig afgepast in de juiste mark-space verhouding zou genereren zolang je de paddle met de wijsvinger ingeduwd houdt, dan heb je daar een hoop gemak van. Het wordt dan ook makkelijker om een bug en een keyer afwisselend te gebruiken, omdat ze dan dezelfde bediening van de paddle vereisen.

Dat gaan we in dit artikel doen en de snijactie van de superlange duw naar links op de paddle, die in keurig afgepaste strepen wordt verdeeld, is een keurslager-actie; vandaar de naam van het apparaatje 'Bug Butcher' die ik ervoor heb bedacht.

Ontwerp

De streeplengte moet drie maal de puntlengte worden. De puntlengte wordt bepaald door de plaats van de gewichtjes op de pendel en is dus variabel. De streeplengte moet automatisch in de verhouding 3 op 1 volgen. Je moet dus de puntlengte voortdurend meten om dat te kunnen realiseren.

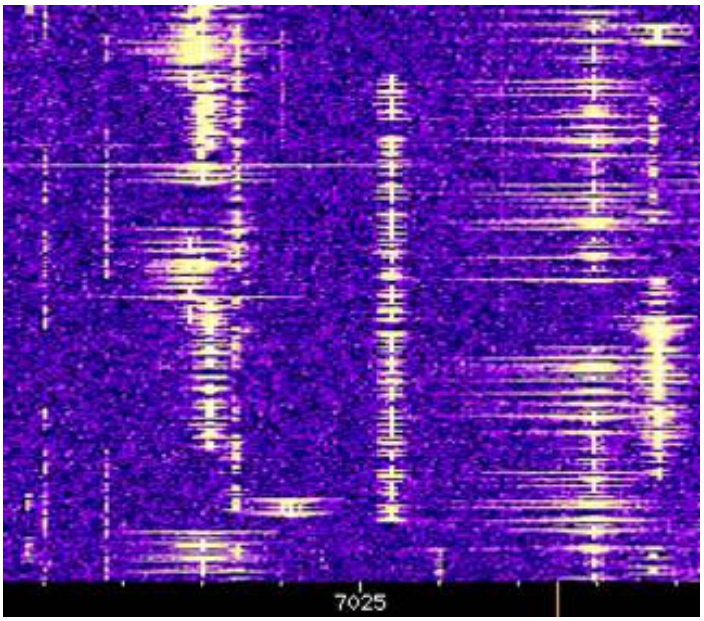
Blijkt de gesloten contacttijdsduur langer dan 2 punten, dan wordt de streep afgemaakt, inclusief de bijbehorende spatie. Is het bug-contact daarna dan nog steeds gesloten dan volgt een volgende volledige streep.

Het heeft een negatieve invloed op je bug-fist als

je verplicht wordt je laatste streep korter dan 3 lang te houden om een volgende ongewenste streep te voorkomen. Dit wordt voorkomen door het testmoment, dat bepaalt of er nog een streep moet komen, op het begintijdstip van de volgende streep te leggen.

Bounce

Een mechanisch contact geeft bounce of contactdenderen. Dat wil zeggen dat zowel bij maken als verbreken van het contact er een aantal snelle maak en verbreekacties bijkomen. Als je daar een eenvoudige zender mee sleutelt heb je kans op flinke bandbreedte ten gevolge van sleutel-click. Bij een dure jappenbak trouwens ook, zo blijkt als ik het spectrum bekijk dat de Websdr ontvanger van de TU-Twente (fig. 1)



toont op een willekeurig weekeinde dus gedurende een of andere contest in de CW band.

Voor de werking van de Bug Butcher zou dat trouwens helemaal rampzalig zijn, want die ziet het snelle schakelen gedurende bouncing als zeer korte punten en maakt de streep dan driemaal dat bedrag lang.

Dat kan vermeden worden als we de uitgang van de Bug Butcher, die de zender sleutelt, in schakelen op de eerste maak van het bouncende sleutelcontact, snelle onderbrekingen daarna niet op reageren, en de uitgang uit zetten op de eerste verbreekactie van het inmiddels langer gesloten contact. Leading en trailing edge wordt dan bounce vrij, zonder de tekenduur aan te tasten, Theoretisch hebben we dan wel een snelheidslimiet; de punten kunnen immers niet korter worden dan de veronderstelde maximaal optredende bounce-tijd. maar dat konden ze toch al niet.

Uitwerking van het ontwerp

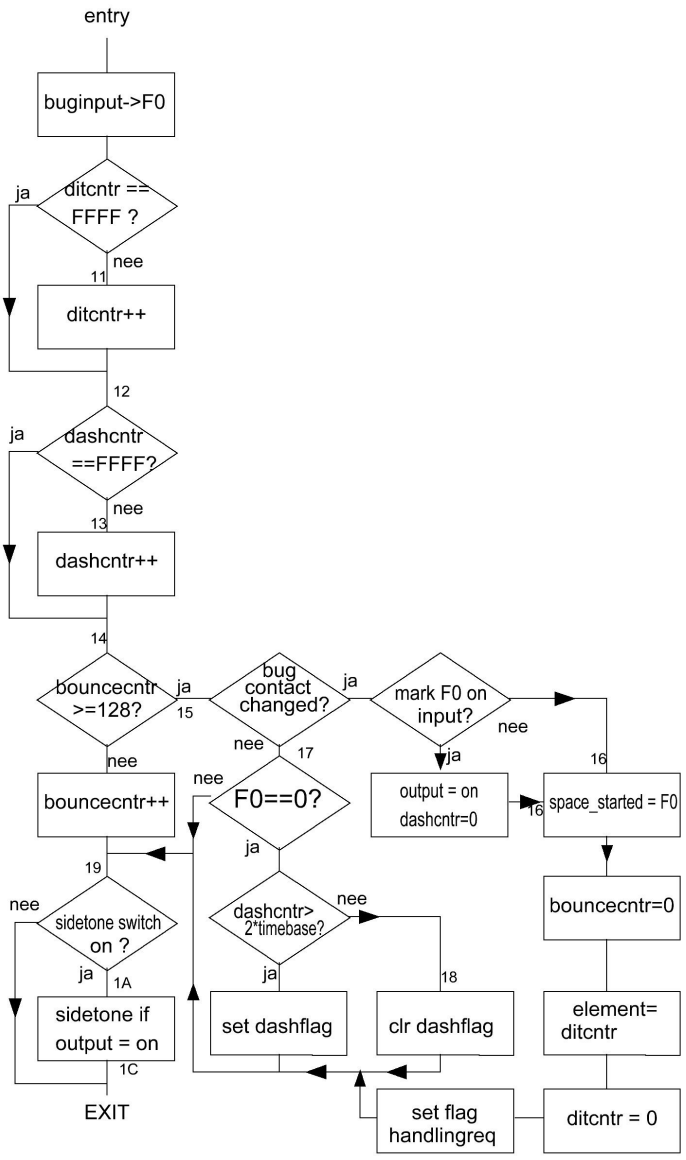
Gebruik wordt gemaakt van de hier in een laatje op voorraad liggende microcontroller AT89S8253 die ik ook in andere ontwerpjes heb gebruikt, welke op mijn website http://pa0wv.home.xs4all.nl/zelfbouw.html te vinden zijn.

Ik programmeer die in assembler. Assembler is een programmeertaal die per type chip verschilt, niet handig dus, maar het voordeel is dat je precies weet wat er gebeurt en de chips dus optimaal kunt gebruiken.

Nu kun je een programma ook omzetten in een andere chip, een Arduino, een Raspberry Pi of zoiets, indien je de flow chart beschikbaar hebt. Die flow charts teken ik er dus bij in dit artikel, en ik leg de werking uit, zodat je zelf, indien gewenst, kunt experimenteren met een andere processor en een andere programmeertaal. Per slot van rekening hebben we onze machtiging of registratie verkregen om te experimenteren en onszelf te ontwikkelen, dat wordt maar al te vaak vergeten door 5nn tu contesters en dx-jagers met jappenbakken.

De werking

Een timer0 veroorzaakt ruim 7800 keer per seconde een interrupt iedere keer als hij van 255 naar 0 flipt. Dat is namelijk de kristalfrequentie/6/256 keer per seconde. De afhandelingsroutine van die interrupts (zie fig. op volgende bladzijde) neemt de stand van de bug op, mark of space; verhoogt een 16 bits ditduurteller 'ditcntr' zolang die niet op zijn maximum 65535 is beland na ruim 8 seconde; en verhoogt tevens een een-byte brede bouncetijdteller, mits die laatstgenoemde beneden de maximale bouncetijdcount staat en doet dan verder niks, buiten het maken van een sidetone als de output op mark staat. Staat de bouncetijdteller echter wel reeds op zijn toegelaten maximum dan wordt die niet verder verhoogd; en in dat geval wordt per sample (bemonstering) gekeken of het sleutelcontact gewijzigd is van laag naar hoog



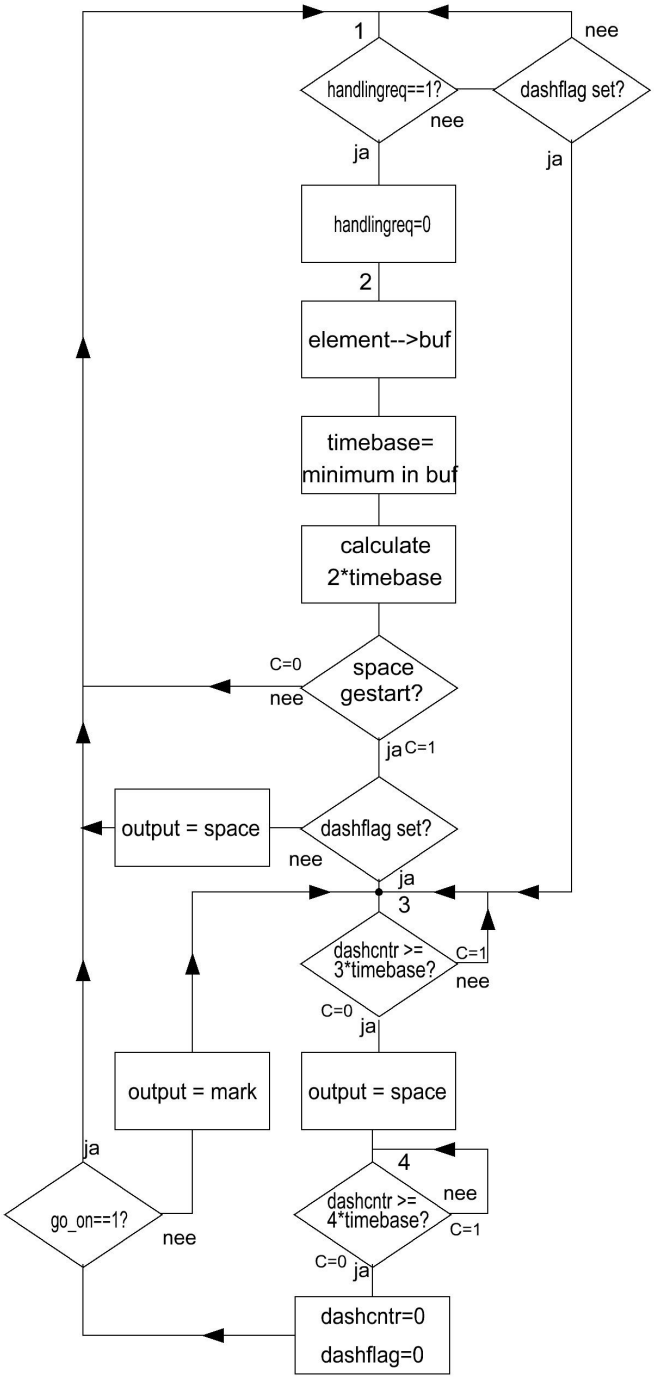
Interrupt handler

(open) of omgekeerd (gesloten).
Is dat het geval dan wordt de bouncetijdteller op 0 gezet en de ditduurtellerwaarde bewaard in 'element' om in het hoofdprogramma 'main' te vergelijken met de laatste acht bewaarde ditduurtellerwaarden, en de kortste van die verzameling wordt als tijdbasis gebruikt om de lengte van de strepen te bepalen. Die 8 waarden kunnen marks of spaces zijn, want van elk tekenelement wordt met ditcntr de lengte bepaald. De ditduurteller in de afhandelingsroutine wordt bij bugcontactwijziging dan onmiddellijk daarna gereset omdat er een nieuwe mark of space is begonnen. Tevens wordt een vlag genaamd space_started geset als het bugcontact in stond en naar uit is gegaan, andersom wordt die vlag gereset. Is de sleutel evenwel niet van positie gewijzigd, meestal dus als je 7800 keer per seconde kijkt, dan gebeurt er ingeval een space bezig is verder niets. Is er echter een mark aan de gang, dan wordt gekeken of de mark inmiddels langer dan 2 keer de tijdbasis duurt. Zo ja dan wordt een dash flag gezet, omdat in dat geval een dash moet worden afgemaakt, ook als het bugcontact te vroeg wordt losgelaten. De ditcntr wordt bij elke sleutelwijziging weer op 0 gereset om de lengte van de volgende mark of space te meten. Er loopt echter ook een dash counter die doorloopt, om in main te kunnen bekijken of de volledige streeplengte is bereikt en de bijbehorende spatie erachteraan. Daarna wordt in main die counter gereset. Na elk tekenelement wordt in de interruptroutine een vlag geset 'handlingreq', het programma 'main' bewaakt die vlag en ziet er aan dat het laatst gemeten tijdsduur in 'element' klaar staat en behandeld kan worden. Als de afhandelingroutine vaststelt dat er een mark is begonnen, wordt de output, die de zender bedient, laag gemaakt.

Het programma main (fig. volgende blz) bewaakt die vlag, als hij geset is wordt hij door main gereset.

Main pakt dan de gemeten elementlengte uit 'element', dat kan een mark of space zijn, zet die in de 8 positiebuffer op de plek van het oudste element, zodat altijd de tijdsduur van de laatste 8 elementen beschikbaar staan. Vervolgens wordt het kortste van die 8 bepaald en opgeborgen in 'timebase'. Tevens wordt 2, 3 en vier keer dat bedrag berekend en in variabelen opgeborgen. Nodig om de strepen te construeren bij aanhoudend indrukken van de bugpaddle in de streeprichting. Na 8 verzonden tekenelementen, waartoe zowel marks als spaces worden gerekend, is die dan uit de buffer verdwenen en doet niet meer mee, dat is van belang als je de bug op een lagere snelheid zet met het pendelgewicht.

Een 'dit' is in dit verhaal gedefinieerd als de kortst voorkomende mark of space. Als we de bug tot 50 wpm willen kunnen gebruiken, is de minimale ditduur 24 ms. Daaruit volgt voor de maximaal toegelaten bouncetijd dat de bouncetijdteller nooit boven 187 kan komen bij een bouncetijd van 24 ms. 128 is een makkelijke te behandelen waarde om als toegelaten maximum te nemen wat wel ruim voldoende zal zijn. Bij een test , waarover verderop meer, blijkt dat vermoeden correct, alleen als je gaat seinen met twee krokodilklemmen tegen elkaar tikken gaat het



main

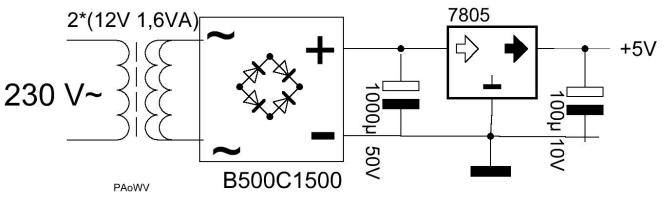
fout, een gewone sleutel en bug geeft geen enkel probleem.

Nu treedt er een moeilijkheid op. Als je de streep indrukt en dat blijft doen, wordt die mark niet afgemaakt, dus komt er geen handlingreq uit de interrupt routine. Daarom draait in de interruptroutine ook een dashcntr mee die daar nooit gereset wordt,
en ingeval er een mark wordt geseind, wordt daar met die dashcntr bijgehouden of die mark inmiddels langer dan 2 dits duurt dat is twee maal de gemeten en vastgestelde tijdbasis, en zo ja dan wordt in de interruptroutine een dashflag geset. In main wordt niet alleen gewacht op de handlingreq vlag voor het geval er een element klaar is, maar wordt ook de dashvlag bewaakt, en zodra die geset is in de interruptroutine doordat de paddle langer dan 2 dits is ingedrukt, wordt die dah afgemaakt en gevolgd door een bijbehorende spatie. Daarna wordt gekeken of de paddle nog steeds in staat en zo ja, dan wordt de dahcounter gereset op 0 en volgt tevens een volgende volledige streep.

Ingeval van een overgang van space naar mark, wordt in de interrruptroutine de output op mark gezet. Terugzetten op space gebeurt in main.

Is het lopende teken een mark en langer dat twee dits, dan wordt de output in main pas teruggezt op space als de dashteller in de interruptroutine drie dittijden heeft bereikt. Als die teller drie dits bereikt, een volle streep dus wordt de output afgeschakeld en bij het bereiken van het einde van de vierde dit wordt gekeken of de input laag is (streep van de bug nog steeds ingeduwd), zo ja dan wordt de dashteller gereset en dit verhaal herhaald om de volgende streep te maken. Dit gaat door zolang de wijsvinger de paddle op de strepenstand houdt.

In de interruptafhandelingsroutine wordt ook gekeken of de output aan is, zo ja dan wordt een sidetone in de vorm van een blokgolf gegenereerd als pin P3.5 , de sidetone schakelaar, hoog is, anders niet. Een versterker stuurt een luidspreker aan die de sidetone laat horen. Volume is instelbaar met een potmeter, die ik als trimpot op de print heb gemonteerd. Schema van controller en versterker op de volgende pagina, het schema van de voeding hieronder.



Voeding

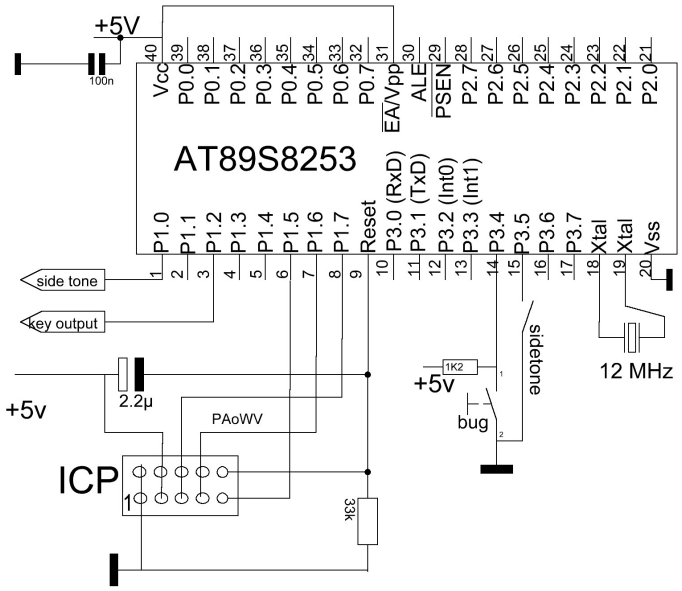


fig 1 Controller en periferie

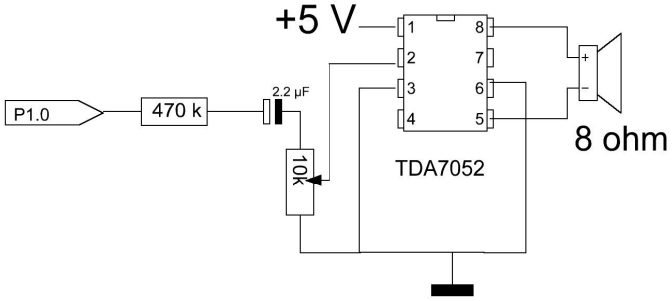
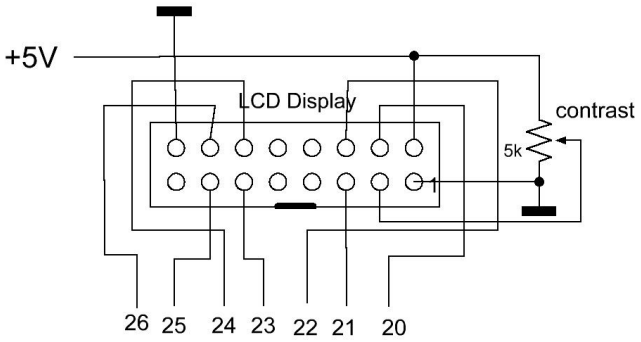


fig 2 Audioversterker

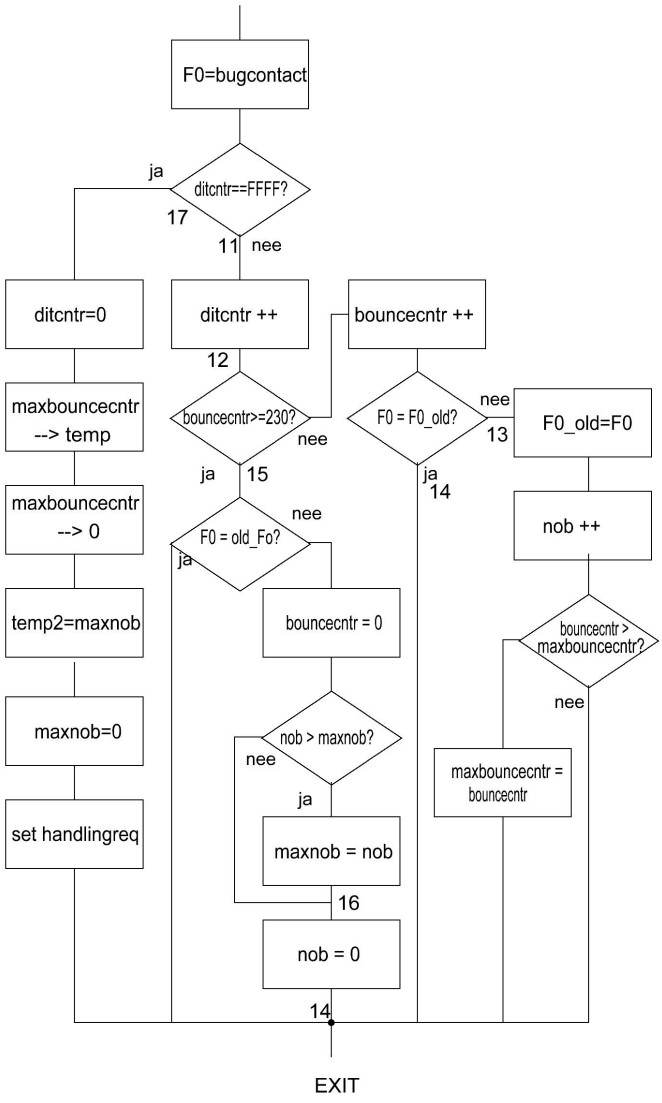
Test van de apparatuur

Het apparaat is geconstrueerd voor gebruik met een bug. Je kunt hem ook testen met een K1EL keyer
(http:\\pa0wv.home.xs4all.nl\pdfbestanden\K1EL bug.pdf) door die in mode 'bug' te zetten. Dat gaat als volgt: Inschakelen netspanning, wachten op R,
cmd knop indrukken wacht op R geef een K met de paddle, de keyer meldt dan de ingestelde mode. bijvoorbeeld KB voor Iambic B mode. Raak de streep paddle herhaaldelijk aan om door de modes te scrollen. Als je de V hoort weer op de cmd knop drukken, dan staat hij in bug mode. De keyer meldt succes weer met een R. Punten dan dus automatisch en streep zolang je die paddle indrukt.

Alles werkt niet direct zoals gewenst, om te debuggen heb ik er een LCDisplay aangehangen via een 16 pins boxed header op port 2. Routines die ik heb gebruikt voor het opsporen van fouten staan onder diagnostics in het programma opgenomen.



Ik heb ook een alternatieve interruptroutine geschreven , flowchart hieronder, als diagnostic die het doel heeft gedurende 7 seconde de sleutelwisselingen met hun bounce te meten, het



Diagnostic int1

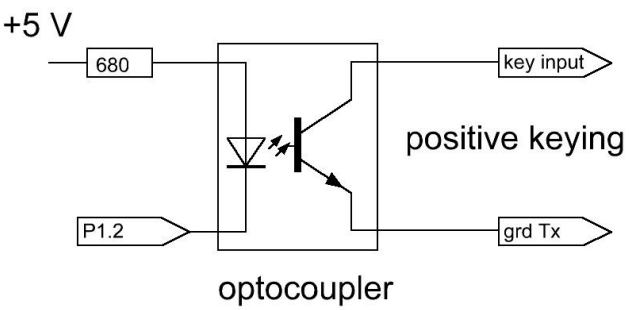
aantal bounces dat optrad bij elke sleutelwisseling wordt bewaard evenals de tijdsduur in de vorm van de hoogste count van de bouncecounter waarbij die optrad. wordt het maximum dat optrad voor bewaard. Voor elke periode van 7 seconde wordt van al die maxima het maximum bepaald en afgedrukt op de LCD. Gewone sleutels geven 1 of 2 bounces, maar ga je sleutelen door twee draadjes tegenelkaar te tikken dan krijg je wel hoge getallen. Die gekozen 128 van de bouncetijdsduur (16 ms) is dus inderdaad aan de zeer veilige kant.

Nu het programma klaar is en goed werkt is uiteraard die LCD en de aansluiting op de processorport P2 niet meer nodig. De connector heb ik er niet afgesloopt, die is onnodig maar blijft gewoon verder zitten.

De connector ICP in het schema is ook uitsluitend nodig om de controller te programmeren, tijdens het ontwikkelen van het programma en als je een geprogrammeerde chip hebt dus overbodig.

Sleutelen van de zender

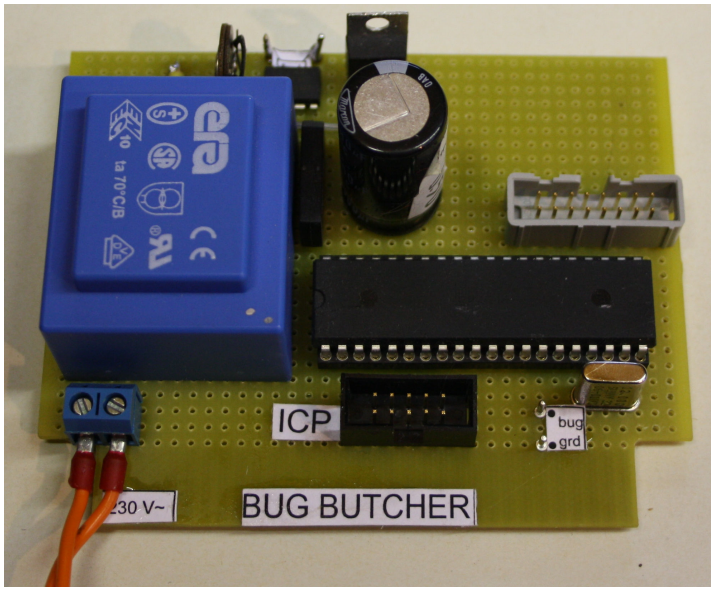
Dat kun je op drie manieren doen. Met tussenschakeling van een reed relais of elektronisch. In dat laatste geval moet je weten wat de open spanning is op een seinsleutel als die niet gesloten is. Positief of negatief ten opzichte van massa, dat is belangrijk te weten, en vervolgens hoeveel stroom er loopt als de sleutel gesloten is.
Getekend schema geeft een mogelijke schakeling als je een positieve spanning naar aarde schakelt . Uiteraard moet de transistor in de optocoupler die spanning kunnen verdragen en de stroom bij een mark.



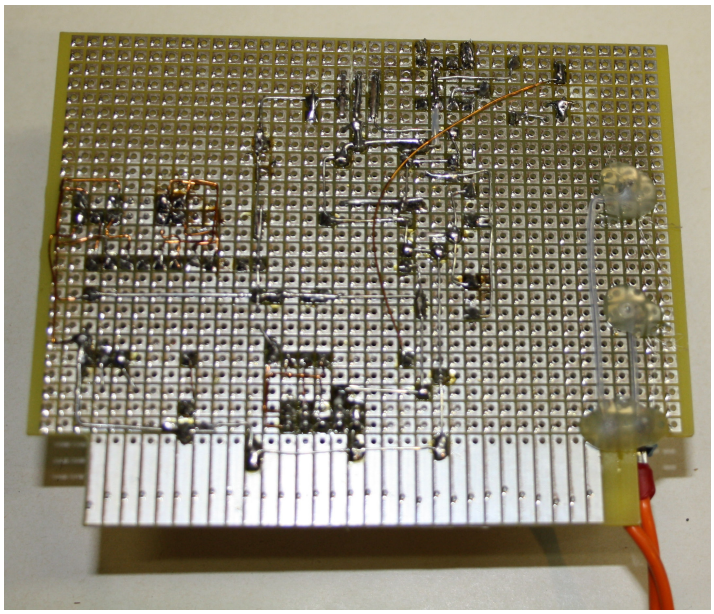
Tx-interface

Bouw

De zaak is op gaatjesprint gezet en doorverbonden aan de achterzijde doorverbonden met draadjes 0,4 mm posijndraad of blank montagedraad.



Als je een schakeling eenmalig maakt is een dergelijke werkwijze naar mijn smaak het beste. Printen zijn voor massafabricage, die zijn niet geschikt, want je kunt niets of nauwelijks iets wijzigen, dus experimenteren is dan taboe.



Nabouwers kan ik een microcontroller voor programmaren, dat kost je inclusief porto 15 euro. Neem indien je dat wilt contact op via email met mijncall@amsat.org, waarbij je mijncall uiteraard dient te vervangen door

PA0WV