MKB

Wim Kruyf PA0WV
pa0wv@veron.nl

Met het MKB kunt u met een AT keyboard morsetekens voor bijvoorbeeld de transceiver genereren.

Inleiding

Het MKB is niet het Midden- en KleinBedrijf dat Japanse koopdozen en antennes in blisterverpakking levert maar een Morse KeyBoard. Voor een ander project kwam ik op het idee om een oud PC-AT keyboard te gebruiken voor het genereren van morsetekens. Dat zou ik 'even' doen maar dat viel zwaar tegen. Uiteindelijk is het gelukt en daarom hier een apart verhaal over dat keyboard. Je kunt een en ander gemakkelijk nabouwen. Het schema is simpel en de geprogrammeerde chip kun je bij mij verkrijgen. De bedoeling is echter dat er ook van kan worden geleerd en dat als je het anders wilt doen met bijvoorbeeld een ander type controller, dat dat ook kan met de verstrekte gegevens en wat zelfwerkzaamheid. Men hoort verhalen dat echte hams een paddle gebruiken en geen keyboard. Leuk maar als je 50 of 60 wpm wilt seinen moet je van goede huize komen. De lieden die het kunnen, prefereren een keyboard omdat het opnemen van QRQ machineschrift stukken prettiger gaat dan van handgegenereerde tekst. Er is een Italiaan op YouTube die de gebruiksaanwijzing van een Begali met 60 wpm zit te sleutelen. Het klinkt afschuwelijk. Het is beslist geen pretje om naar te luisteren. Als je niet overtuigd bent lees dan eens het verhaal van de echte QRQ jongens die 'rag chewen' met snelheden die soms oplopen tot 120 wpm op de website van W4BQF
http://sites.google.com/site/tomw4bqf/copyingcwover70wpm .

PC-AT keyboard

Op diverse plaatsen op internet zijn gegevens te vinden over het PC-AT keyboard die cruciale fouten bevatten, voorts elkaar op punten tegenspreken, onvolledig zijn en bij toepassing dus kunnen leiden tot lastige problemen. Het keyboard wordt gevoed via de DIN 5-pens aansluitconnector en mijn eerste exemplaar, een oud exemplaar uit 1984, trekt bij 5 V een niet-onaanzienlijke stroom van 120 mA. Daarom heb ik een moderner exemplaar ingezet dat bovendien meer toetsen heeft. De keyboards met PS-2 connector zijn hetzelfde behalve de connector waar een verloop voor verkrijgbaar is. Er zijn naast de 5 volt voeding een datalijn en een kloklijn op de connector aanwezig. De klok wordt altijd door het keyboard gegenereerd. De datastroom kan naar het keyboard toe zijn, het kunnen commando's zijn en uiteraard van het keyboard af zijn. De richting van de informatie en de dataflowcontrol worden bepaald door de controller van het aangesloten apparaat. Beide lijnen, klok en data, zijn open collectoruitgangen en zijn in rust hoog. De controller van het op het keyboard aangesloten apparaat kan ze laag trekken. Het keyboard zendt alleen als de kloklijn niet laag getrokken wordt door de controller. De kloklijn fungeert dus tevens als CTS (Clear To Send). Het keyboard genereert de klok en ontvangt een commando als de datalijn eerst laag getrokken wordt door de controller RTS (Request To Send) tot de kloklijn een serie klokpulsen begint af te geven. Er is altijd een startbit, een stopbit, 8 databits en een odd parity bit. Bij verzenden van een byte naar het keyboard toe geeft het keyboard aan het einde van het verzonden teken een laag bit af ter bevestiging. Je kunt (commando's) sturen, bijvoorbeeld om een LED in het keyboard aan of uit te doen of om het laatst ontvangen byte te herhalen. In het geval van een verzonden commando wordt na dat commando tevens een byte 0xFA gestuurd als bevestiging, of 0xFE als het verkeerd overkomt of als het commando niet kan worden uitgevoerd; dat na elk byte van een meerbytes commando. Tot zover de details. Om 'porten' van de basisroutines get-byte en put-byte op een andere controller mogelijk te maken heb ik de flowcharts van die assembly routines getekend. Voor degenen die er echt meer van willen weten zijn er diverse flowcharts en tabellen op de website van *Electron* website [1] geplaatst, zeer de moeite waard. Uiteindelijk heb ik de zaak goed kunnen uitzoeken en werkend gekregen middels een geheugenoscilloscoop die eenmalig ontvangen signalen in een geheugen vasthoudt en periodiek als een normaal oscilloscoopbeeld weergeeft. Het keyboard geeft zogenaamde scancodes af zowel bij het maken als bij het loslaten van een toets. Die kunnen uit meer bytes bestaan, maximaal 8. Met tabellen worden die in de controller omgezet in ASCII, de toetsen die geen ASCII zijn zoals de Fn toetsen, pijltjes enzovoorts worden als upper ASCII (bit 8 = hoog) volgens een in principe door mij willekeurig gekozen code afgegeven zodat die toetsen ook ergens anders voor gebruikt kunnen worden en dat in dit ontwerp ook worden. In de vertaalroutines van scancode naar ASCII wordt bijgehouden of er sprake is van capslock, alt, shift, control en numlock. De bijbehorende LEDs worden met com-

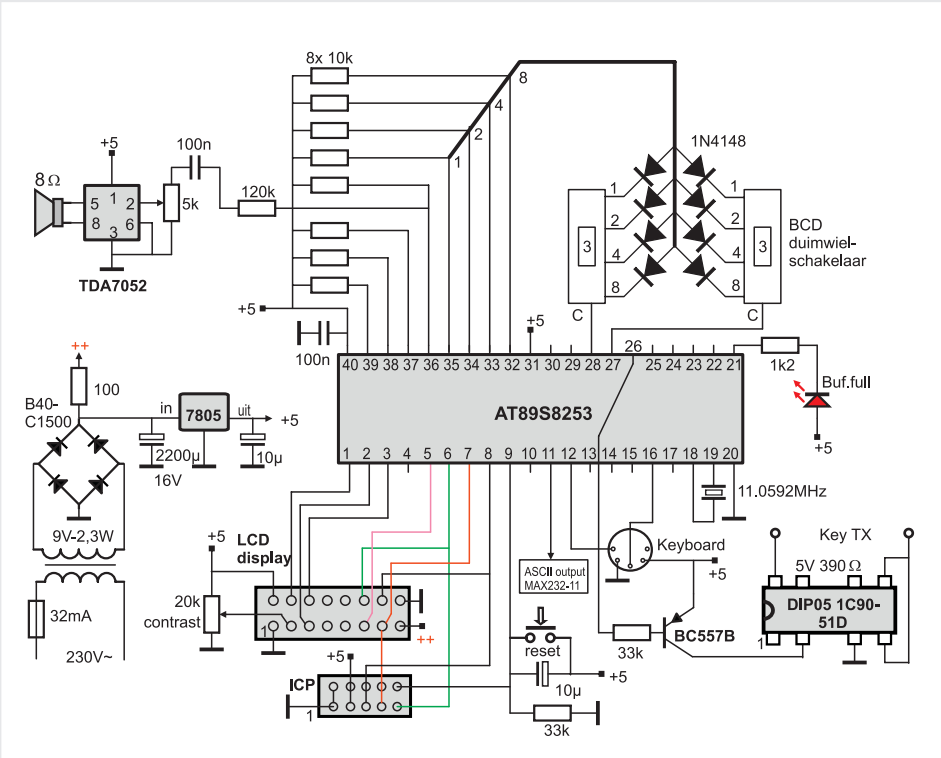

Fig. 1 MKB schema

mando's naar het keyboard toe bediend. Als er fouten in de keyboardcommunicatie worden gedetecteerd, wordt daar door hervragen adequaat op gereageerd. Helpt hervragen (drie keer) niet dan volgt een commando: keyboard reset. Het keyboard zelf bevat een buffer voor scancodes die vooruittypen mogelijk maakt en die bij een keyboard-reset leeg wordt opgeleverd. Die buffer is echter te klein om bij sneller typen dan zenden dit foutloos af te handelen; het zijn namelijk 16 bytes en die kunnen de scancodes van maximaal 5 letters bevatten. Daarom worden in de controller de ingetypte letters in ASCII code in een circular buffer met een capaciteit van 80 bytes opgeslagen. Tamelijk veel maar dat biedt de gelegenheid tijdens een QRS rag chew QSO tijdens de uitzending van de zenderbeschrijving tussendoor even weg te lopen om koffie te gaan drinken. De inhoud van de buffer die nog niet is uitgezonden staat op een LCD. Backspace heeft tot gevolg dat in de buffer, indien niet leeg, het laatste karakter wordt weggehaald. Op die wijze kun je dus misslagen in de ingetypte tekst corrigeren. De delete toets op het 'nummerpad' rechts, laat hele woorden tegelijkertijd verdwijnen. Die wist namelijk tot hij een spatie tegenkomt. De morsetekens zoals <SK> dus' ...-.- ' worden prosigns genoemd. Voor de 7 prosigns zijn aparte fonts ontworpen en die worden bij de initialisatie in de LCD geladen. Raakt de buffer voor meer dan 87 % vol dan gaat er een rode LED branden om te waarschuwen dat er bij doortypen dataverlies dreigt. Dat kan natuurlijk ook een zoemertje of iets dergelijks zijn. Zie ook het schema in figuur 1. Verder kun je in geval de ontvangst van BK tijdens een QSO de buffervoorraad data onderbreken (met de Pause/Break toets op het toetsenbord) of zelfs geheel weggooien met ctrl-alt-del dat de vertrouwde combinatie vormt om de hele zaak te resetten. Bij een reset wordt wel de ingestelde snelheid bewaard in EEPROM. De morsesnelheid is instelbaar tussen 6 en 99 wpm door Alt en dan twee cijfers in te toetsen terwijl je Alt ingedrukt houdt. Het is verstandig tijdens dat gebeuren even de pauzetoets in te drukken want de cijfers schuiven achter elkaar op hun plaats zodat na het eerste cijfer de snelheid een onverwacht zeer hoge waarde kan hebben. De morsesnelheid is in woorden per minuut volgens de PARIS-standaard. Die wordt dan ook in EEPROM gewijzigd zodat de volgende keer na inschakelen van de netspanning het keyboard op de laatstgebruikte snelheid terugkomt. Het blokje van 6 toetsen, tussen het alfanumerieke linker deel van het keyboard en het nummerpad in, is gebruikt om de prosigns in te coderen. Dat zijn de tekens SK, AR, etcetera. Die toetsen kun je van een etiketje voorzien tot je ze uit je hoofd weet. Of BK een prosign is of niet kan zelf bepaald worden door de letters b en k apart in te toetsen, of door de pijl omhoog onder het prosignblok, die bk als zevende prosign bevat. Er is een 4-cijferige contestteller geïmplementeerd. Wordt op de grote + toets rechts op het nummerpad gedrukt dan komt die teller in de zendbuffer te staan en wordt tevens daarna met 1 verhoogd voor de volgende keer. De - toets erboven verlaagt de teller met 1. Voor het geval het getal herhaald moet worden is die functie noodzakelijk. Bij een reset wordt de teller op 0001 geïnitialiseerd. Je kunt de teller op een willekeurige gewenste stand zetten door die in de lege buffer in te tikken en uitzending te voorkomen met de pauzetoets. Vervolgens de entertoets op het keypad indrukken, dan verdwijnt het 4-cijferige getal in de teller. Ter bevestiging verdwijnt het van de display. Verdwijnt het niet dan waren het geen 4 cijfers en kan het naar wens alsnog gewijzigd worden. Een en ander is goed uitgezocht zodat een betrouwbaar geheel is ontstaan. Onder de F toetsen kun je een bericht opnemen van maximaal 128 bytes per toets. Dat kan als volgt gebeuren:

1. Druk de pauzetoets in;
2. Type de gewenste tekst in en kijk op de display of die naar wens is; zo niet dan aanpassen met backspace en Del;
3. Druk Alt in en vervolgens tevens de F toets waar het bericht onder moet komen.

Je kunt vervolgens kiezen of je pauzetoets bedient en het bericht uit de buffer uitzendt of dat je dat bericht in de buffer met backspace, delete of in een klap met ctrl-alt-del wist. Druk je dan op de F toets dan komt dat bericht direct in de buffer te staan. Je kunt het ook meermalen achter elkaar in de buffer zetten als daar geen plaats voor is. Het is mogelijk per QSO op die manier de call van het tegenstation er in te zetten, dan hoef je die tijdens het QSO niet herhaaldelijk in te tikken. Met een paar van dergelijke macro's tik je met een paar toetsaanslagen zo een standaard hello-goodbye-QSO bijelkaar in je buffer. Dan krijg je de bekende macro-QSO's met hams die zodra je iets afwijkends vraagt of doet schielijk QRT gaan, al of niet met de smoes dat ze BCI of TVI hebben of dat zojuist de zendbuizen beginnen te smelten, danwel, erger nog, schoonmoeder zojuist gearriveerd is. Dat kun je overigens ook in een macro zetten. Je kunt ook een bericht onder een Fn toets als volgt aanpassen:

1. Pauzetoets indrukken om uitzending te verhinderen;
2. De betreffende Fn toets indrukken, dat geeft het eronderliggende bericht in de buffer en dus op de display;
3. Edit het bericht;
4. Plaats het met Alt Fn terug in de EEPROM.

Na de vertaling tot ASCII die dat deel van het apparaat universeel toepasbaar maakt voor andere doeleinden en daarom de ASCII output op de UART van de controller naar buiten voert, wordt een volgende vertaalslag gemaakt waarbij de keuze uit morse en Hell code is geïmplementeerd. Je kunt van morse naar Hell schakelen met de linker en rechter pijltoets op het pijltoetsenveld onder het prosign blok.

Ontwerpproblemen

De flowcharts van get-byte en put-byte zijn aanvankelijk als polling routines gerealiseerd; prima voor een terminal die ASCII code via een UART afgeeft. Nu kun je echter hier niet de output van het keyboard in de controllerbuffer krijgen en daarbij voorlopen op de morse die de buffer leeghaalt en die ook met polling van een interrupt gedreven tijdteller werkt die de lengte van strepen en punten bepaalt. Een oplossing zou kunnen zijn twee controllers te gebruiken: een voor het keyboard die via de SSI of UART ASCII afgeeft en een tweede die de ASCII met zijn UART aanpakt en in de circular buffer zet waar dan ook de morse of Hell verwerking in zit die de buffer leeghaalt. Om niet direct op deze onelegante aanpak terug te vallen is om te beginnen de polling van het keyboard vervangen door een interrupt service routine. Elke klokflank van het keyboard geeft dan een interrupt die als taak het volgende keyboardbit opneemt. Je hebt dan dus 11 interrupts per keyboard byte. Dat volstaat echter niet om de buffer sneller te kunnen vullen met de keyboard-output dan dat de morse hem aftapt. Om dat te bereiken zou gestrooid in de morseroutines een subroutineaanroep om het keyboard te servicen meervoudig moeten worden opgenomen; ook verre van fraai. De morse hapert als je commandotoetsen bedient zoals capslock of numlock want afhandelen van commando's betekent wachten op de acknowledge bytes, parameter bytes ingeven etcetera. Daarom heb ik eerst gedacht aan twee stacks. Er draaien in de controller dan twee aparte programma's: het ene haalt het keyboard leeg met polling en geeft ASCII af aan de buffer in het tempo waarin getypt wordt en het andere programma haalt naar behoefte die buffer in zijn eigen tempo leeg om de ASCII output te transformeren naar Hell of morse code met de gekozen snelheid en verzorgt de display-verversing. Een interrupt routine die optreedt bij een timer overflow en die tevens de sidetonefrequentie bepaalt, drukt dan als extra taak alle gebruikte registers van het ene programma op diens stack en dan komt het: het verwisselt de stackpointer met een tweede stackpointer en gaat dus bij einde interrupt de registerinhouden van de tweede stack halen en die zijn van het tweede programma. Iedere keer als die interrupt optreedt worden dus de stacks verwisseld. Dat is een context switch. Beide programma's komen dan even vaak en even lang aan de beurt en hebben beide nagenoeg de halve beschikbare tijd als timeslice beschikbaar. Voor elk van de programma's lijkt het alsof er een interrupt optreedt die ruim de helft van de tijd in beslag neemt. Op die wijze lopen er twee programma's in de controller onafhankelijk van elkaar, om beurten. De duur van de tijdsleuven wordt bepaald door de frequentie van de interrupt. Dat kan echter in dit geval niet werken omdat elke tijdsleufduur gebruikt voor de context switch en het andere programma te lang is om binnen 25 microseconde van een keyboardklokflank de datalijn

gegarandeerd te sensen, iets dat moet. Nu kun je dat sensen echter met een andere externe interrupt doen; de klokflank van het keyboard veroorzaakt dan een externe interrupt en onafhankelijk van de even of oneven timeslice wordt die onmiddellijk geserviced. Enige uitzondering is dat de context switch interrupt net aan de gang is maar die duurt kort. Dan lijkt dat fout te gaan omdat de interrupt de registers opbergt op de verkeerde stack als toevallig het morsedeel aan de gang is maar dat geeft niks want die interrupt wordt altijd eerst afgemaakt. Dat wil zeggen dat hij de registers die hij opborg op een willekeurige stack ook weer van diezelfde stack haalt. De keyboard interrupts lopen dan altijd door ook als het andere proces dat de morse genereert op het moment van interrupt aan de gang is. Dat is de wijze waarop een en ander is geïmplementeerd.

Testen

Dit testen is niet bij nabouw maar bij het porten naar een ander type processor of tijdens het ontwerpen van de schakeling van belang. De circular buffer wordt getest door die te vullen met keyboardaanslagen tot hij vol is (carry set) en vervolgens te legen tot hij leeg is (weer carry set). Het onder interrupt schrijven van de put-byte en get-byte van het keyboard kan apart worden getest. Het kost dagen om onverwachte fouten op te sporen. De vertraging van de 'Aha erlebnis' groeit blijkbaar exponentieel met het vorderen van de leeftijd van de ontwerper. In casu zet je voor put-byte de data klaar om onder interrupt bitsgewijs verstuurd te worden en je mag niet met het get-byte voor ontvangst van het acknowledge byte beginnen voordat die data daadwerkelijk verstuurd is. Twee wachtlussen dus: een voor toegang tot het put-byte en vervolgens een na het put-byte om te wachten tot de data effectief geheel verstuurd is, wat blijkt uit het feit dat de bitcounter van 11 naar 0 gedaald is. Open deur, maar ja... Enfin, na een halve fles port geconsumeerd te hebben zag ik het ineens. Als ik weer een weerbarstig probleem heb dan is dus op basis van deze ervaring weer naar die oplossingsmethode te grijpen. Bij het dual procesontwerp wordt het tweede proces eerst heel eenvoudig gemaakt, namelijk keer steeds een poortpen om van polariteit. Hang je daar een oscilloscoop aan dan zie je dat dat gebeurt gedurende 100 µs en dat er 150 µs niks gebeurt op die poortpen. Daaruit is te concluderen dat beide processen 100 µs van de 250 µs beschikbaar hebben en dat de contextswitches 50 µs van die tijdspanne dus 20% van de totale processortijd consumeren bij die schakelfrequentie. Zo snel schakelen hoeft niet: 2 ms lijkt me snel genoeg. Een punt of streep van morse kan dan in het nadeligste geval 2 ms te lang worden. Bij 99 wpm, een morsesnelheid die door sommige hams in de wereld met keyboards naar eigen zeggen gehaald en op het gehoor genomen wordt, duurt een punt 12,5 ms en die wordt dan dus maximaal 16% te lang. Van de morsetx is bekend dat je dat niet merkt. Minder vaak context switchen wil ik niet doen, mede omdat de sidetone die nu 250 Hz is dan te laag in frequentie wordt. Als ik me kwaad maak kan ik dat met een derde proces oplossen maar ik maak me niet kwaad. Timer 1 wordt daarom voor de context switch ingericht met een herhalingstijd van 2 ms, omdat een timer default in mode 1 van de MCS-51 architectuur door 8.192 deelt en dus geen aanpassing van het deeltal en dus tijd vergt gedurende de context switch, uitgezonderd nu het 'on the fly' setten van het meest significante bit van de lopende teller. Opletten dat beide programma's in de controller niet elkaars data wijzigen. Dat leidt tot onverwachte effecten. Dit werkt en na zoveel gedoe geeft dat aangevuld met de high makende harsdampen van de soldeerbout de vreugde die slechts de zelfontwerper/bouwer kan ervaren.

Hell

In de Hell mode levert de buffer voor elk erin staand ASCII teken een Feld Hell teken met een snelheid van 122,5 baud. Dat bepaalt de bandbreedte. De leesbaarheid wordt bij ontvangst verhoogd als je die pulsen van ruim 8 ms of een geheel veelvoud daarvan kunt verplaatsen over 4 ms. Ze worden dan niet smaller, de bandbreedte neemt dus niet toe. Dat betekent hier dat elke letter bestaat uit kolommen van 14 bits waarvan altijd elke zwart- en wittijd ten minste twee bits duurt. Alle letters zijn als hoofdletters in een tabel opgenomen van 64 letters maal 12 bytes. Dat is genoeg voor 6 kolommen per letter, de zevende kolom is altijd een lege letterspatie. Testen gebeurt door het audiosignaal van de sidetoonspeaker aan te sluiten op de geluidsingang van de laptop waarop het programma Hellschreiber v4.0 van IZ8BLY [1] draait.

Nabouw

Het keyboard zelf zit in een plastic kast en het niet afgeschermde aansluitsnoer werkt

Foto 1 Het MKB

als antenne. Dus als je veel HF in je shack hebt krijg je problemen. Daar kom je als SSB'er achter als je je lippen aan de microfoon brandt. Ferriet kan je wellicht redden, een C'tje van de klok- en datalijn naar aarde misschien ook. Ik heb de zaak ongevoelig gemaakt voor netschakelaarpulsen etcetera door bij een interrupt van het keyboard te controleren of de spanning dan ook daadwerkelijk laag is en verder door een netfilter toe te passen. Of dat nodig is weet ik niet: het lag voor € 1,- op een vlooienmarkt. Ik heb het gekocht en wil het dan wel toepasssen. De DIN connector heeft op de soldeerlippen gezien linksom draaiend de pennummers 1, 4, 2, 5 en 3. Dat lijkt een rare volgorde maar dat komt omdat er ook 3 pens DIN-connectors zijn die de plaats van de pennen 1, 2 en 3 bepalen. Pen 1 is de kloklijn, pen 2 de datalijn, pen 4 is ground en pen 5 is +5 volt. Het geheel is gezet op een half euro-

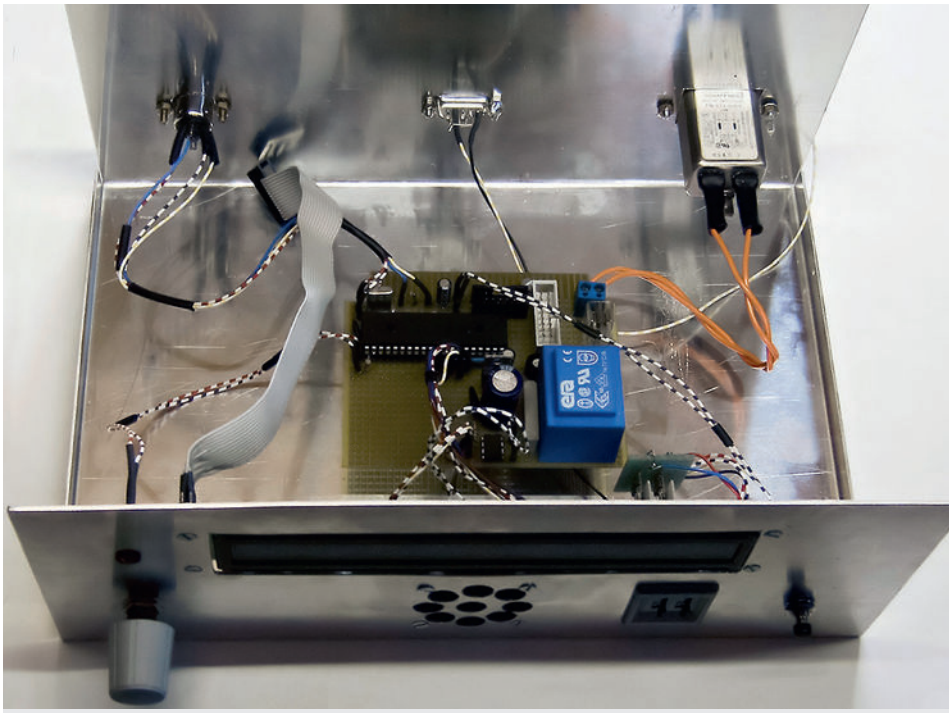

Foto 2 Inkijkje in het MKB controllerkastje met de halve eurocard

formaat. Dat is niet de grootte van een 50 ct munt, gaatjesprint, voedinkje erbij en een goed verkrijgbaar kristal van 11,0592 MHz erbij. De controller kun je geprogrammeerd bij mij bestellen via pa0wv@amsat.org voor € 15,- inclusief porto en verpakking. Het kastje is gekocht voor minder dan € 10,- bij Baco. Het is al omgezet; zagen en boren zijn dus ronduit lastig. Het 2-regelige display met 40 karakters per regel is van Samsung (zonder back-lite) en is geleverd door Rein Pentina PA0RKP die altijd op vlooienmarkten te vinden is. De aansluitkabel heb ik naar de achterzijde verplaatst. Dat is door de doorplatering een lastig karwei. Denk er om dat dan de even en oneven nummers van de kabel van plaats verwisselen. Als je geen behoefte hebt aan ASCII output op de RS232 connector dan kun je die connector en de MAX232 met elco's gewoon weglaten. De ICP voet kan ook weggelaten worden als de zaak niet in-circuit geprogrammeerd hoeft te worden omdat je een geprogrammeerde controller chip monteert. De foto van het apparaat en het schema tonen duimwielen voor de snelheidsbepaling. Die kunnen worden weggelaten uit het schema en het apparaat omdat de snelheid gemakkelijker en goedkoper ingesteld wordt met het toetsenbord zoals hiervoor aangegeven. Degenen die toch BCD duimwielen willen toepassen voor de snelheidsinstelling, kunnen dat bij bestelling van de chip melden, dan wordt die daarop geprogrammeerd. Alle verdere onderdelen zijn vlot verkrijgbaar en niet kritisch qua opstelling, grootte etcetera. Eerst de voeding maken, monteer een voet voor de controller, meet tussen pen 20 en pen 40 of er 5 volt staat, dan de controller erin.

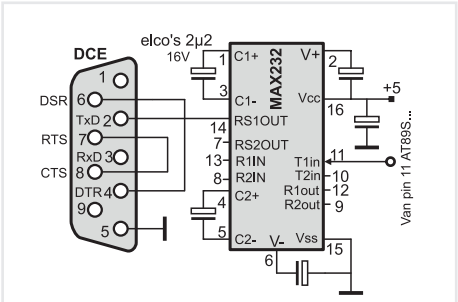

Fig. 2 Schema van de eventuele RS232 interface

Foto 3 Vooraanzicht van het controllerkastje

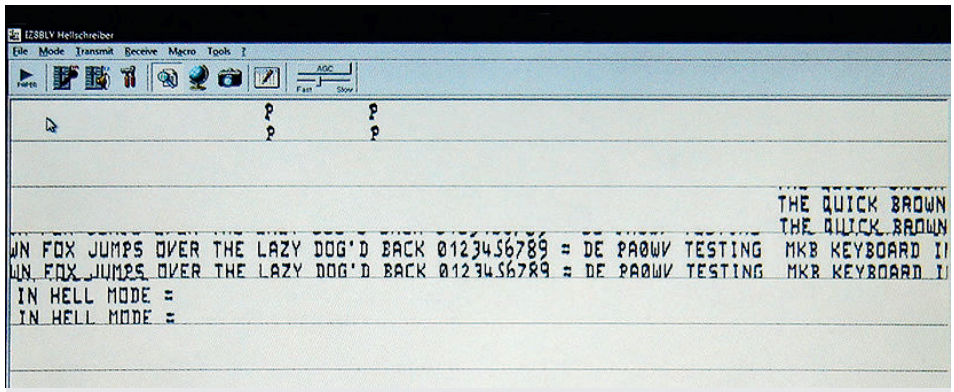

Foto 4 Het Hell testresultaat

Opletten dat er geen pennetjes dubbelklappen en dat hij niet een halve slag gedraaid in de voet wordt gezet. Dat de controller werkt kun je in geval van twijfel meten op pen 30. Daar staat dan een signaal van 3 MHz. De LCD contrastregelaar moet je zo instellen dat je beeld hebt. Het audio kun je testen door met je vinger te drukken op pen 2 van de TDA7052. Ik heb een reed relais gemonteerd met potentiaalvrije contacten. Daarmee kun je elke normale zender sleutelen maar je kunt natuurlijk daarnaast of in plaats daarvan ook een transistordriver monteren, eventueel met een optocoupler voor negatief of positief schakelende seinsleutelaansluitingen. Hoe je zender sleutelt kun je meten op de contacten van een aangesloten seinsleutel door daar de open spanning en de kortsluitstroom te meten. 55 es bcnu in CW.

Indien in plaats van de, niet meer leverbare, TDA7052 een TDA7052A gebruikt wordt moet voor een juiste werking pen 4 niet direct maar via een 100 k weerstand aan aarde gelegd worden.

Referentie

[1] http://electron.veron.nl