Een programmer voor de chipcorder ISD2560

door Wim Kruyf PAoWV

De chipcorder ISD2560 kan op vele manieren in de shack gebruikt worden. Wim PAoWV beschrijft in dit artikel een programmer voor dit IC met verrassende toepassingsmogelijkheden als een QSO-machine voor de Ham Radio de Super Luxe phone boys, die gewoon doorgaat met QSO'en en logboek invullen als de operator even wat anders aan het doen is.

De chipcorder is een IC familie, die analoog geluid kan opbergen en vasthouden en naar believen afspelen.
De 2560 chip kan 60 seconden bevatten met een sampling rate van 8 kHz; er is ook een 120 chip, die in feite evenveel geheugen (480 kB) heeft, maar met de halve bemonsteringssnelheid en navenant mindere geluidskwaliteit de dubbele speelduur heeft. De chips kunnen aan elkaar geregen worden, zodat als de eerste is uitgepraat de tweede verder gaat.

Werking chip

Het opgenomen geluid wordt in analoge monsters bewaard (100 jaar). De kwaliteit is equivalent met 8 bit bemonstering, dus een S/N ratio van 48 dB. Voor opname is er een microfooningang met versterker en AVC ingebouwd, die een regelbereik van 39 dB heeft. Bij opname is een 5 pool anti aliassing filter met 3,4 kHz grensfrequentie ook intern aanwezig. De klok is ook intern beschikbaar.
Helemaal compleet dus, je kunt er zelfs direct een luidspreker aanhangen, maar het volume laat dan wel te wensen over, namelijk 12 mW in 16 Ω. De voedingsspanning is 5 volt.

Het leuke van de chip is dat hij hex 258 = 600 adressen bevat, die de 480 kB spraaktekst verdeelt in stukjes van 0,1 seconde.
Deze adressen zijn extern aanstuurbaar en er wordt in die mode dan afgespeeld vanaf dat adres. Het geluid houdt op als er een marker wordt gevonden; die markers worden automatisch bij opname geplaatst als je een pauze inlast. Je kunt de chip dus aansturen op een adres en hij speelt dan af tot de volgende marker.

Toepassingsmogelijkheden

Een toepassingsvoorbeeld is de eerder in CQ-PA maart 2007 beschreven SGIYE, die telwoorden bevat en waarmee je praktisch tot oneindig kunt doortellen met slechts 22 geluidsfragmenten, die maar een gering deel van de beschikbare geheugenruimte in beslag nemen.
Andere toepassingen zijn sprekende snelheidsmeter in je auto, verjaardagskalenderklok, BigBen bim-bam klok (een nostalgisch geluid, omdat dat een halve eeuw geleden het ANP nieuwsbulletin altijd voorafging); een lees/praatplankje en een leer-klokkijken-klok voor kleuters, een QSO machine voor de Ham-Radio-de-Super-Luxe phone-boys, die gewoon doorgaan met QSO'en en logboek invullen als ze afwezig zijn.
Maar ook bijvoorbeeld inbraakalarm dat de inbreker vermanend toespreekt als hij je shack betreedt; enfin noem maar op, toepassingen in DOKA's waar je geen hand voor ogen ziet, sprekende universeelmeter, zodat je voorkomt dat je op een display of schaal moet kijken en ondertussen de meetpennen kortsluiting laat maken, een speller om je CW snelheid te verhogen en last but not least: toepassingen die een hulp voor visueel gehandicapten in onze hobby zijn. Je kunt het allemaal verzinnen.

Beperking

De moeilijkheid ligt in het programmeren. Je kunt dat met de hand doen, door een geluidssignaal aan te bieden waarvan je bij het begin een schakelaar moet indrukken die chip enable bedient en op het einde ook voor het plaatsen van de marker. De adreslijnen van de chip A0-A9 moeten dan vast ingesteld staan op hex 340 tijdens programmeren in die push button mode.
Als je dat doet, dan gaat het een paar keer fout. Maar alles went (volgens de XYL: zelfs een vent), echter er is een belangrijk bezwaar: namelijk dat die fragmenten door die handbediening praktisch nooit op de zelfde adressen in de chip zullen beginnen, zodat je van een controller die alles aanstuurt per geprogrammeerde geluidschip de adressen zal moeten wijzigen.
Elke controller heeft dus doorgaans andere adressen geprogrammeerd en dat is vervelend als je zoiets wilt dupliceren voor nabouwers. Je kunt dan niet alle controllers hetzelfde programmeren, omdat alle geluidschips verschillen.
Moduleer je per ongeluk over, of druk je de markerknop aan de late kant in na afloop van het fragment, dan kun je ook weer helemaal opnieuw beginnen en dat is allemaal een nogal tijdrovend gedoe.

Er is wel een adresmodus dat de chip zonder externe adresaansturing het n-de geluidsfragment pakt, als je n startpulsen geeft. Dat komt aan dat bezwaar tegemoet, maar dan krijg je wel het hele fragment en verlies je de mogelijkheid het aan de voorzijde wat in te korten, omdat daar wat teveel spatie zit. Ook woorden gedeeltelijk afspelen van halverwege tot het einde gaat dan niet meer. Het is me overigens ook na diverse pogingen niet gelukt die cueing mode aan de praat te krijgen.

Oplossingen

De vraag is of aan dat tijdrovende programmeren met de hand wat te doen is. Dit artikel beschrijft een poging die met succes in die richting gedaan is. Voor SGIYE chipset bestellingen heeft het minder voordeel, want al die wav-bestanden die ingestuurd worden zijn anders en er wordt maar één chip per bestand geprogrammeerd.
Voordeel is wel dat je met wat geluk de chip in een keer goed kunt programmeren. Vergissingen en het vaststellen van de beginadressen van de geluidsfragmenten zijn dan minder tijdrovend. We zullen zien waarom. Een ander belangrijk voordeel is, dat je de beschikbare geheugenruimte op de chip volledig kunt uitbuiten, omdat geen opnameduur verloren gaat aan pauzes voor en na het geluidssignaal.

Samenvatting van de voordelen van het programmeren boven handprogrammeren:

1. Er gaat meer audio op de chip dan met handprogrammeren.
2. Ongewilde pauzes voor en na het woord, die storend kunnen zijn bij het gebruik als woorden aan elkaar gelijmd (geconcateneerd) worden, worden geëlimineerd.
3. De beginadressen direct na de geplaatste markers worden automatisch vastgesteld door de programmer in een sterk verkorte procedure en bewaard in EEPROM van de programmer ter overneming in de assembler source listing van de te programmeren controller en die zijn naar wens later en herhaaldelijk uitleesbaar via de LCD display van de programmer.
4. Testen van de gevonden adressen gebeurt ook in de programmer, zodat je direct weet of de programmering geslaagd is en de woorden dus niet in tijd verschoven zijn en dan dus een stuk aan het begin of einde missen.
5. Eenmaal gemeten tijden kunnen bewaard blijven in de programmer EEPROM in schakeltijdbestanden, zodat je die niet opnieuw hoeft in te brengen op een later tijdstip. Maximaal 7 bestanden van 256 bytes kunnen worden bewaard.
6. Schakeltijdbestanden kunnen langer zijn dan 256 bytes, dat bleek noodzakelijk in de praktijk en is later als modificatie aangebracht.

7. Uit de aanwezige schakeltijdbestanden kan worden gekozen; ze zijn gekenmerkt met een 10 byte naam op de LCD display.
8. Uit de schakeltijdbestanden wordt tevoren berekend hoeveel opnametijd van de chip zal worden verbruikt. De totaaltijd wordt in ms op de LCD display weergegeven.
9. Gemeenschappelijke geluidsfragmenten met twee verschillende beginadressen kunnen met de programmer getest worden op het beste startadres van het korte fragmentdeel.

De programmeerprocedure

Een geluidsbestand (wav-file) moet je altijd beschikbaar hebben, dat is te maken met een microfoon en een geluidskaart in de PC en Soundblaster Wave Studio die ik daarvoor gebruik. De geluidsfragmenten moeten er allemaal opzitten, fouten mogen ertussen zitten mits elders gecorrigeerd, en spaties mogen er ook zijn, zodat de totale speeltijd van de wav-file de 1 minuut ruim overschrijdt, geen probleem.
Je kunt trouwens met Wave Studio altijd stukken uit de file wegknippen om hem naar wens in te korten, of spaties opnemen als je een woord in twee stukken wilt splijten. Dat laatste blijkt inderdaad nodig te zijn, omdat de op te nemen fragmenten niet al te dicht op elkaar mogen zitten.

Als je twee woorden nodig hebt bijvoorbeeld "morgen" en "overmorgen", dan hoef je alleen maar "overmorgen" in te spreken want als je dat woord op een hoger adres start is het vanzelf "morgen" geworden. Dat spaart dus chipgeheugen, zodat je er meer woorden op kunt plaatsen. Dat grapje werkt uiteraard niet in de cueing mode, die een geluidsfragment verderop pakt bij elke stuurpuls op CE (chip enable).

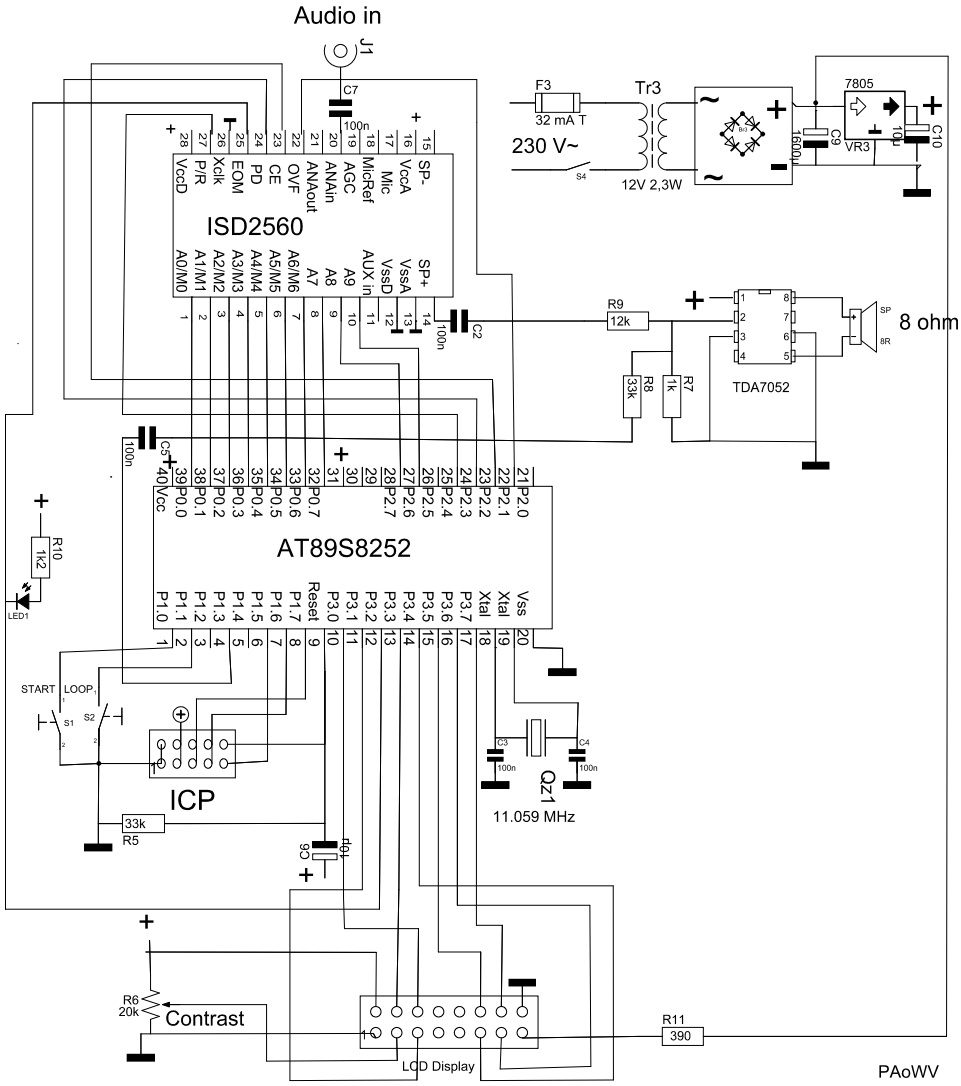
Die wav-file laad je dan in Wave Studio en in Wave Studio kun je dan fragmenten afkaderen met een cursor precies een fragment dat in de chip moet en later door de controller adresseerbaar moet zijn. Met Shift P kun je dat afspelen om te controleren. Is dat fragment naar je zin dan noteer je de door het programma op je scherm opgegeven starttijd en eindtijd in milliseconde op een vel papier, of die type je direct in je PC.
Dat herhaal je voor alle op te nemen fragmenten, zodat je met een lijst eindigt van start- en stoptijden in milliseconde nauwkeurig.
Die lijst wordt ingebracht in het EEPROM van de programmer, waar dit artikel over gaat.

Ontwerpoverwegingen bij de bediening

Ik heb ervan afgezien om dat inbrengen van start- en stoptijden, wat vrijwel allemaal 5 of 6 cijferige getallen zijn, via drukknoppen of andere bedieningsorganen direct in de programmer te doen. Het is namelijk mogelijk om met een PC de lijst te editten en te assembleren en in EEPROM te zetten van de controller via de ICP, de InCircuit Programming connector, voorzien van een filename. Je kunt dus achteraf uit maximaal 7 ingebrachte files met start- en stoptijden kiezen welke je wilt programmeren. Maximaal kunnen er in de 2 kB EEPROM 7 stuks van 256 byte, waarvan je er altijd een in onbruik geraakte dan kunt vervangen door een nieuwere als de zaak vol zit.
Het bleek in de praktijk nodig om langere lijsten te gebruiken, een entry bevat namelijk 6 bytes voor de aan- en uittijden, zodat er dan slechts 40 geluidsfragmenten opgenomen kunnen worden. Door een wijziging van het programma kan een lijst nu meerdere secties van 256 bytes bevatten. Theoretisch kun je maximaal 300 verschillende korte geluidsfragmenten in de chip zetten. Dat bleek uit proefnemingen, zodat je dan de volledige capaciteit van 7 maal 256 bytes nodig zou hebben, je komt dan uit op een limiet van 296 korte geluidsfragmenten die opgenomen kunnen worden met de programmer.

De programmer bevat een controller Atmel 89S8252 en een te programmeren ISD2560. Die laatste kan er het beste met een 28-pens DIL ZIF voet worden ingezet. Dat is zo'n voet met een handeltje waarmee je de IC-pennen vastklemt.
De controller zet dan de chip in de programmeermode. De controller bedient via een output-pen de chip enable en de bedoeling is, dat hij op de opgemeten tijden in de wav-file de geluidsfragmenten opneemt, terwijl de hele wav-file achterelkaar afspeelt.
Er zit een (rode) startknop op de programmer. Druk je die startknop in, dan is dat het tijdstip 0 en dan gaat de controller precies op de ingeprogrammeerde tijden de CE herhaaldelijk bedienen. Het ligt voor de hand dat je gelijktijdig de wav-file moet gaan afspelen op t=0 anders klopt het niet. Dat gebeurt met handbediening van de computer en de startknop, die "gelijktijdig" worden ingedrukt. Dat is een zwak punt, want dat leidt soms tot verschoven woorden die dus stukjes missen.
Nu is als 70 plusser mijn reactietijd niet meer die van een 20 minner, wellicht dat anderen daar minder problemen mee hebben.

Ik opperde de gedachte om nog eens met een solenoide op de keybordtoets te gaan proberen de gelijktijdigheid te verbeteren, maar dat leek me eigenlijk teveel moeite. Ik voelde er meer voor om een harde ge-



luidspuls aan het begin van de wav-file te zetten, die gelijkgericht de startknop in de vorm van een relais bedient. Dat is beter reproduceerbaar. Dat het relais daarna op de rest van het geluid gaat staan klepperen speelt geen rol, want eenmaal gestart gaat hij door tot het bittere einde van het schakeltijdenbestand. Na enige ervaring bleek echter dat de nauwkeurigheid ruim voldoende is als je de geluidsfile start met Shift P; vervolgens na korte tijd gelijktijdig Shift P en de startknop indrukt.
De computer herstart dan onmiddellijk het afspelen van de file en heeft geen last van hinderlijke onvoorspelbare vertragingen in dat geval.
Ook als 70 plusser met een tragere reactietijd kun je met twee handen nog wel twee knoppen gelijktijdig indrukken. Dat gaat in de praktijk zo goed, dat elke behoefte om genoemde verbeteringen aan te brengen nagenoeg zijn verdwenen.
Mocht er toch iets fout gaan dan is de chip onmiddellijk na de test in de programmer weer te programmeren. Kost geen verdere moeite of hinderlijk tijdverlies. Blijkt er een tamelijk vast tijdverschil tussen de schakeltijden en de geluidsfragmenten, dan kun je een stiltefragmentje op de ms nauwkeurig voor de geluidsfile plaatsen of weghalen.

Een ander punt is dat de ISD 2560 een interne klok heeft. Er lopen geen twee klokken gelijk, heb je dus een volgende 2560, dan loopt die wat trager of sneller, zodat je uiteindelijk wellicht op andere adressen uitkomt. De specsheet spreekt van 2 à 5% spreiding als temperatuur- en exemplaarspreiding.
Dus het beste is gebruik te maken van de mogelijkheid een externe klok aan te bieden uit het kristal van de controller bij het programmeren, die klok moet 1024 kHz zijn, die heb ik niet voorhanden en ik ben niet van plan identieke series te programmeren, dus dat heb ik zo gelaten.
Anders moet je het controllerkristal van 11,059 MHz vervangen door een kristal van 12288 kHz (handelswaarde) dat na deling met een HCMOS chip 74HC92 door 12 de gewenste 1024 kHz levert. Dat kan ook met een 11 deler op het normale kristal van 11,059 MHz, echter dan iets minder nauwkeurig.
Dan kom je namelijk uit op 1005 kHz en dat is binnen 2% van de nominaal gewenste waarde, zodat je nog geen problemen kunt krijgen met het interne klokgestuurde anti-aliassingfilter of reconstructiefilter. Je verliest dan alleen een half procent van de beschikbare opnametijd. Dat is pakweg een kwart seconde, verwaarloosbaar dus.

Is de zaak geprogrammeerd dan kun je de programmer in de afspeelmode zetten, hij speelt dan door de aangesloten luidspreker fragment voor fragment af ter controle. Dan kun je horen of de zaak niet overstuurd is of te zwak opgenomen. Er is daarom een 1 watt versterker TDA7052 op de print erbij gezet, die een 8 Ω speakertje van 10 cm diameter aanstuurt.

Het vaststellen van de fragmentadressen in de ISD2560

Is het programmeren en controleren gebeurd, dan is de vraag op welke adressen de geluidsfragmenten beginnen. Normaal bood ik bij programmeren met de hand dan achtereenvolgens alle adressen aan met een testroutine in de controller op de doelschakeling en noteerde die adressen waarbij een fragment begon. Dat is ook tijdrovend en als de adressen met morse worden afgegeven omdat het een schakeling zonder display betreft, op den duur ook storend voor huisgenoten. Zeker als je meerdere chips achtereenvolgens gaat programmeren. Pas na het seinen van pakweg 500 adressen van 4 hextekens heb je de gewenste gegevens bij elkaar.

Nu kan dat automatisch gebeuren, namelijk als je op een adres start dan krijg je een fragment te horen tot de eerstvolgende marker, start je een adres hoger dan is het fragment wat korter want er ontbreekt een stukje geluid van 0,1 seconde aan de voorzijde, enzovoorts, totdat het ineens weer langer is, omdat je de marker gepasseerd bent en het volgende fragment afspeelt.
Welnu de adressen waar de speelduur ineens langer wordt dat zijn de beginadressen van de fragmenten. De controller van de programmer probeert dat, en noteert de adressen die een begin van een fragment zijn in zijn geheugen. Daar wordt altijd het achtste stuk van 256 bytes uit het EEPROM voor gebruikt. Dat kan dan dus maximaal 128 ISD2560-adressen van elk 2 byte bevatten.

Heb je een volgesproken chip met laten we zeggen n woorden of geluidsfragmenten, waarbij n een geheel getal is, dan kost dat natuurlijk wel nagenoeg evenveel tijd als het op het gehoor bepalen van de adressen. Het verschil zit alleen in de tijd die nodig was om de adressen in morse te zenden via de luidspreker.
Omdat er 600 adressen zijn en n woorden, heb je dat er gemiddeld per woord 0,5*600/n*(600/n+1) adressen worden afgespeeld en voor alle n woorden samen dus ongeveer 300*(600/n+1) adressen. Per adres zit je op 60/600 = 0,1 seconde, de totale onderzoektijd voor het bepalen van de beginadressen wordt dus voor n woorden 30*(600/n+1), wat bij benadering 18000/n seconde is. Bij 22 woorden is dat ongeveer een kwartier. Als je slechts 2 berichten van 30 seconde opneemt is de onderzoektijd 9000 seconde en dat ligt in de buurt van 2,5 uur. Niet helemaal eerlijk berekend, want het laatste woord in de chip hoef je niet meer te meten. Niet ongeduldig worden en op bedieningsknoppen gaan rammen dus, was dan het devies.

De vraag is of je wat aan dat ongerief kunt doen. Dat kan en is in deze programmer gerealiseerd.
Als je met een geluidsfragment begint meet de controller tijdens het afspelen de tijdsduur van dat fragment op de milliseconde nauwkeurig, dus je kunt vervolgens een sprong in de adresruimte maken tot dicht voor het einde van het fragment. Dat bekort de zoektijd in theorie in het voorgaande voorbeeld van 2,5 uur tot 60 seconden per volgeprogrammeerde chip, onafhankelijk van het aantal woorden op de chip. Dat idee heb ik dus geïmplementeerd. Dat werkt nu allemaal volautomatisch.
In de praktijk bleek dat de routine doorzoekt naar de laatste marker en dan ging de zaak op tilt. Dat heb ik opgelost door tevoren met de controller het aantal geluidsfragmenten te tellen dat in de schakeltijdentabel is opgenomen, en vervolgens het zoeken naar markers te stoppen en die zoekroutine te verlaten met het plaatsen van een end of file marker FFFF in het markerbestand dat ook in EEPROM in het achtste 256 byte segment wordt opgebouwd door de zoekroutine en bewaard blijft tot het overschreven wordt bij een volgend gebruik van de markerzoekroutine. Als het aantal markers dat gevonden is gelijk is aan het aantal geluidsfragmenten stopt deze routine dus.

Definitieve testfase

De vastgestelde adressen worden vervolgens gebruikt om de chip ter controle van achter naar voor af te spelen. Daarmee wordt voorkomen dat het ontbreken van een marker tussen twee geluidsfragmenten niet wordt opgemerkt.
De adressen van de markers die in EEPROM zijn geschreven door de markerzoekroutine in de bovenste 128 bytes van het 8ste filebestand (dat dus minder groot mag zijn) worden dan gelijktijdig getoond.

Heb je woorden opgenomen die je ook als deelwoorden wilt gebruiken zoals “overmorgen” als je ook “morgen” nodig hebt, dan krijg je op grond van bovenstaande keurig het begin van overmorgen als adres beschikbaar, maar waar precies “morgen” begint is dan onbekend. Dat kun je wel gokken, maar dat blijkt niet betrouwbaar. Ook daar is rekening mee gehouden, want als je in het EEPROM met je PC de volgnummers van de betreffende te verdelen woorden in de top laadt, een byte per woord, dus vanaf $7FF en lager, eronder af te sluiten met de EOF delimiter $FF, dan worden die woorden met die volgnummers in de ISD2560 chip elk twee keer afgespeeld met per keer elk adres als beginpunt die tot dat woord behoort. De beginadressen staan op de display, dus je kunt makkelijk vaststellen op welk adres “morgen” in het genoemde voorbeeld precies begint.

Uitleesfase

Als het programmeren en adreszoeken gebeurd is, zijn de gevonden fragmentadressen stuk voor stuk op de LCD gedurende 5 seconden uit te lezen, zodat een bijpassende controller waar je applicatie, die gebruik maakt van de geluids-chip, in moet komen, kan worden geprogrammeerd met die adresgegevens, die vanaf het display handmatig worden overgenomen in de assembler broncode.
In de praktijk bleek dat de waarden 5 seconden op de display laten staan en dan de volgende te tonen, volstaat. Kun je doorschrijven of typen en hoef je intussen niks te bedienen. In principe is het ook mogelijk de EEPROM uit te lezen met de PC, om de adressen te vergaren. Dat heb ik niet gedaan.

In een eerdere fase van de ontwikkeling werden de markers bewaard in het RAM geheugen in een circular buffer. Echter het aantal waarden dat in de buffer past was te gering, door de beperkte geheugenruimte voor de buffer in processor RAM. Dat heb ik aanvankelijk opgelost door niet voor elk markeradres dat opgeslagen moet worden 2 bytes te nemen maar slechts 1 byte.
Het andere meest significante byte is namelijk 0 of 1. Dat kun je voorspellen, omdat het aanvankelijk steeds 0 is, terwijl het tweede byte oploopt voor de adressen en dan, als het tweede byte ineens geringer is, het eerste 1 is geworden. Het is niet helemaal waterdicht, het verstand van de gebruiker speelt een rol om te bepalen of de gepresenteerde bytes van een 0 of 1 prefix moeten worden voorzien. Het gevolg is wel dat je het dubbele aantal adressen kwijt kunt en dat volstond aanvankelijk, maar fraai is anders, vandaar dat ik 256 bytes EEPROM segment waar maximaal 127 markers in kunnen een betere oplossing vond die uiteindelijk in het laatste model dat hier beschreven wordt is geïmplementeerd.

De resterende hoeveelheid RAM waar de circular buffer gebruik van maakte werd trouwens bij het vorderen van het ontwerp aan geknabbeld, zodat ik besloten heb maar gewoon met EEPROM de zaak op te bergen, het blijft dan staan ook als de voedingsspanning eraf is. Ook een belangrijk voordeel.

Het invoeren van de start- en stoptijden van de wav-file

Nu is het tijdens de ontwikkelfase van de programmer steeds inprogrammeren van alle tijden in de assemblerlisting van de programmer zelf, vervolgens assembleren en programmeren ook een handeling die niet in een definitieve oplossing thuishoort. Dat is dus vanzelfsprekend geëlimineerd in het definitieve ontwerp.
Om dat op te lossen kun je de LCD display op het zaakje, die de gevonden markeradressen van de geluidsfragmenten opgeeft tevens gebruikt voor het invoeren van de tijden en kun je met twee knoppen, een voor de cursorpositie en een voor de waarde, de tijden in ms erin zetten. Ik vind dat erg omslachtig en heb dat niet gedaan. Je moet immers ook waarden kunnen editten en tussenvoegen. Nieuwe tijden worden vergezeld van een 10 byte filenaam ingeprogrammeerd in EEPROM, dat kan achteraf gebeuren.
Per bestand wordt 256 byte EEPROM of een veelvoud daarvan gereserveerd, de eerste 10 bytes zijn bestemd voor de filenaam in ASCII, de laatste 3 bytes als end-of file marker FFFFFF en dan resteren dus minimaal 246 bytes. Elke start- en stoptijd neemt 3 bytes, dus een geluidsfragment eist 6 bytes, zodat je 246/6 = 41 fragmenten kunt programmeren. Heb je er minder dan is er dus een end pointer bestaande uit 3 bytes FF, die aangeven dat het laatste fragment opgenomen is. Heb je er meer dan loopt het bestand gewoon door in het volgende EEPROM segment tot de EOF entry FFFFFF wordt bereikt.

De bestanden worden in EEPROM gezet, dus als je dezelfde geluidsfile voor een aantal chips wil gebruiken hoef je dat maar een keer te doen. Tussendoor kan de spanning eraf, de adressen blijven bewaard. Ze kunnen ook ge-edit worden in je PC in de assembler broncode van de EEPROM van de programmer. Voordeel is ook, dat je ze hex in het EEPROM kunt zetten, terwijl je ze in assembler decimaal intypt. Ze zijn decimaal beschikbaar als je ze opmeet in de wav-file in Wave Studio op je PC, dat voorkomt dus gereken en vergissingen. Zet je ze (ter controle) op de LCD display dan worden ze ook decimaal weergegeven. Er kunnen maximaal 7 verschillende adresbestanden gelijktijdig in het EEPROM aanwezig zijn, omdat dat 2 kB groot is, alle oude bestanden blijven dus bewaard zolang het niet vol is, en zijn van een naam voorzien, die je via het LCD scherm kunt selecteren.
Ga je via je LCD een bestandsnaam kiezen, dan worden alle eerste stukken van 10 bytes getoond, dat kunnen echter stukken zijn van langere bestanden, als een bestand meer dan 1 stuk van 256 beslaat. Dat vuiltje heb ik verwijderd door als een bestandsnaam niet uitsluitend printable ASCII bevat het over te slaan.
Abusievelijke keuze van een deelbestand wordt zo effectief vermeden.

De drie byte start- en stoptijden zijn noodzakelijk, omdat twee bytes tekort kan zijn. Dan kun je maar tot 65 seconden wav-file gaan, maar bij 3 bytes ontstaat een overmaat, omdat je dan tot 4 uur lange files stukjes uit zou kunnen kiezen, die samen een minuut geluid maximaal opleveren. Overdreven veel, maar 2 bytes is dus te weinig.

Een routine zet de hex 3 byte getallen in het geheugen om in decimaal met onderdrukte voorloopnullen ter controle, indien gewenst, op de LCD display.

Een menu waar je doorheen kunt rouleren met een van beide bedieningsknoppen - de gele loopknop - biedt via de LCD display alle besproken mogelijkheden aan. De tweede knop - de rode startknop - activeert zo'n mogelijkheid. Bij recording wil je voorkomen dat dit onvoorzien gebeurt; daar wordt dus een mededeling gegeven dat de startknop op scherp staat na een keer drukken. Druk je een tweede keer op de startknop dan gaat hij zonder vertraging van debouncen onmiddellijk opnemen. Bedenk je je echter, dan kun je met de loopknop in dat geval weer terug naar het keuzemenu.

De knoproutine is zo gemaakt dat er doorheen vallen door bouncing niet gebeurt. Kom je binnen, dan wacht hij namelijk 25 ms. Is een van beide knoppen dan nog ingedrukt, dan blijft hij daar hangen in een testlus tot beide knoppen zijn losgelaten; daarna volgt een debouncing van 25 ms en dan pas komt de lus waar de knoppen normaal inzitten, die kijkt of een van beide wordt ingedrukt. De routine wordt dan onmiddellijk verlaten met een carry vlag aan of uit, afhankelijk van welke van beide knoppen ingedrukt is. Dat werkt inderdaad feilloos.
Het is tevoren mogelijk om alle opneemtijden op te tellen om te kijken of die de capaciteit van de ISD2560 tezamen niet overschrijden. Ook dat is een optie die in het menu wordt aangeboden.
In de praktijk bleek het ook nuttig om een adres naar keuze af te spelen: het beginmarkeradres op de display, een woordvolgnummer en de tijdsduur van dat fragment gelijktijdig op de display te krijgen. Ook die optie heb ik dus in het menu opgenomen.

De bouw

Optimistisch dacht ik, dat een half euroformaat gaatjesbord, 10 bij 8 cm dus, genoeg zou zijn. Het gaat erop, maar het is krap aan. Dat komt, omdat ik er ook een netvoeding heb opgezet. Dit omdat stekkervoedingen nogal wat ruimte innemen, die zijn vaak verschillend, maar hebben doorgaans dezelfde aansluitconnector, met het gevaar dat je daarmee apparatuur opblaast door de verkeerde te gebruiken. Met de slankere geschakelde modellen heb ik heel slechte ervaring wat betreft netlek, sindsdien gaat mijn PC gehandicapt door het leven…
Een netvoeding betekent echter dat er ook netspanning op de print aanwezig is, en in mijn geval een niet geïsoleerde zekering, wat het gevaar van elektrocutie inhoudt, of dat je bij onbedoeld aanraken in een spastische beweging de print door de shack

lanceert die daarbij een hoop schade kan aanrichten aan je Ham Radio de Luxe opstelling.
Murphy voorspelt namelijk dat die aldus gelanceerde print het scherm van de laptop naar de Filistijnen transformeert alvorens in de FT dx 9000D in te slaan.

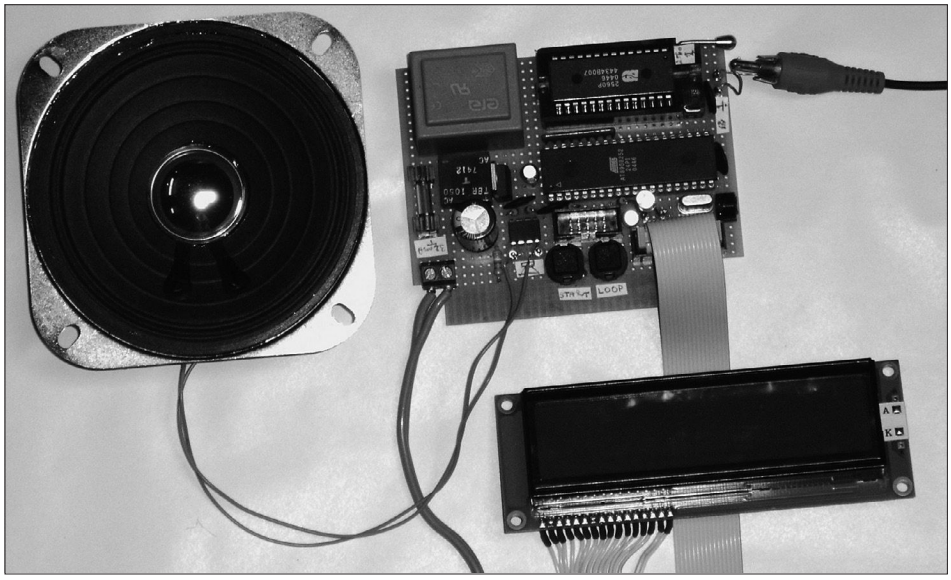
Geïsoleerde zekeringhouder en een stukje printplaat of pertinax tegen het netspanningsdeel aan de bedradingszijde monteren, bezweert dit gevaar goeddeels. Een stuk rontgenfilm van Kodak (tks PAoLQ) is ook een perfecte duurzame isolatiefolie.
Je kunt ook de hele zaak in een kastje monteren, met een gewone 28 pensvoet voor de ISD2560, en daar dan een 28 aderig bandkabeltje insteken dat hem verbindt met de ZIF voet die je buiten op de kast monteert. Wel zo veilig. Ook prettig want de positie van de voet wordt dan niet door de plaats van het hendeltje bepaald, met het prettige gevolg dat hij (180 graden verdraaid) veel makkelijker te bedraden is.
Je kastje, in mijn geval zo'n lessenaarmodel uit de Baco dump (de onderzijde meldt: BimBox no 6007), biedt dan ook plaats aan de LCD display, de bedieningsknoppen en de luidspreker. De praktijk leert dat als je zaken niet behoorlijk inkast en documenteert, het na verloop van enige tijd in de junkbox verdwijnt en gekannibaliseerd wordt.

De trafo is 12V 2,3 W, de zekering 32 mA T, de LCD verlichting wordt daarom een beetje getemperd door die te voeden uit de ruwe plus direct over de elco op de graetz brug, via een weerstand van 390 Ω. Dat houdt de dissipatie in de weerstand ook beperkt zodat een kwart watt model kan worden gebruikt, wat weer ruimte spaart op de print. De kleine displays geven dan nog voldoende licht, de grotere modellen niet meer.

De opstelling van de onderdelen blijkt uit de foto.

Een 10 pins boxed header op de print is voor incircuit programmeren, die blijft erop zitten als een elektronische navel, en is tijdens de ontwikkeling van de software van eminent belang om een beetje te kunnen opschieten omdat je dan stukjes programma makkelijk en snel kunt uitproberen en debuggen. En in het eindmodel ook nog van belang omdat ik heb gekozen voor het programmeren van EEPROM met de start- en stoptijden via de connector. Daarom is hij voorzien van een stukje 10 aderig bandkabel dat aan de achterzijde door de kastwand steekt, zodat het aan de PC interface gekoppeld kan worden voor het laden van het EEPROM.

Bij de pennen 32 t/m 40 is een randje 10 kΩ weerstanden in een SIL behuizing gemonteerd afkomstig van een sloopprint uit



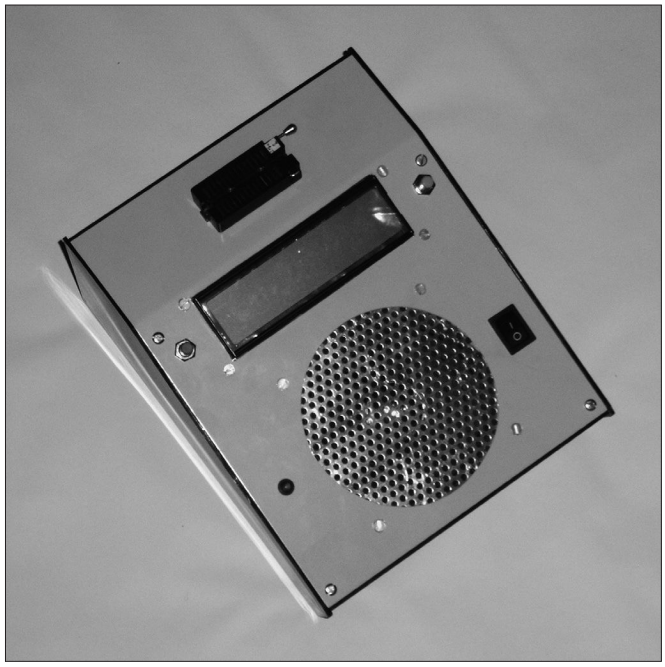
een dumpbak; als pull ups, omdat die port anders (als enige) als je een 1 erin schrijft een zwevende hoogohmige waarde levert.
De audioinput is via een tulp contra chassisdeel niet via de bandkabel naar de chip gevoerd, maar via een afgeschermd stukje snoer direct op de pennen 20 en 13 (analoge aarde) ingevoerd. Een TDA7052 levert voor de controle van het geluid luidsprekervermogen af aan een 8 Ω speakertje van 10 cm diameter.
Het kastje zit aardig vol. Het frontpaneel bevat speaker, LCD display, 2 drukknoppen voor de bediening en een LED.
Ook een netschakelaar, die is van belang, omdat bij het wisselen van de ISD2560 de spanning eraf moet, om onvoorspelbaar gedrag te voorkomen. Op de bovenzijde prijkt de 28 pens ZIF-voet.

De LED is verbonden met de EOM, End of Message signaalpen van de ISD2560, die geeft onder andere het verloop van de opname aan, en is een bruikbaar hulpmiddel om afwijkend gedrag en verschoven opnametijden te kunnen beoordelen.
De achterwand is bezet met de cinch connector voor audioinput, een sleuf voor de bandkabel om de EEPROM te laden en het netsnoer. Het kastje is van kunststof, uitgezonderd de voor- en bovenzijde die van aluminium is, kunststof kan bij alle apparatuur die niet gelijktijdig met zenden in gebruik is, en het bewerkt makkelijk. Een figuurzaag, wat schuurpapier op een stokje, een breekmesje en een boortje is alles wat nodig is om tot het uiterlijk te komen dat op foto 2 wordt getoond.

Testen van de werking en de eigenschappen van de ISD2560

Stukje experimenteel elektronicaonderzoek hoort er ook bij. Met een C programma van een tiental regels heb ik de assemblerlisting gemaakt (van 340 regels) van een tijdenbestand dat vervolgens in het EEPROM van de programmer geprogrammeerd is. Dat bestand bevat achtereenvolgens tijden van 840 ms steeds 10 ms korter tot 20 ms toe. Met dat bestand dat 83 geluidsfragmenten behelst, die onderling steeds een halve seconde gespatieerd zijn, heb ik een continue geluidsfile opgenomen, daar zijn dus hapjes uit op de chip gezet.
Daarna de geprogrammeerde chip al die fragmentjes afgespeeld en dat met een microfoon opgenomen als wav-file op de computer om de afgespeelde fragmenten qua tijdsduur te kunnen onderzoeken.

De programmer netjes ingebouwd in een kast.



Het blijkt met die proef dan dat:

A) De afgespeelde stukjes geluid van 0 tot 20 ms korter zijn dan ze erop gezet werden, de verkorting zit vermoedelijk aan de voorzijde want er is altijd eerst een spike zichtbaar aan de voorzijde.
B) Het niet mogelijk is kortere stukjes geluid dan 120 ms erop te zetten. Bij de kortere gaat de zaak op tilt.
C) Bij het kortste fragment van 120 ms bleek dat de adresmarkers 2 uit elkaar liggen, dat is dus kennelijk de minimumafstand. Dat zou dus duiden op een afspeelduur van 200 ms.
D) Speel ik een opgenomen stukje van 120 ms herhaaldelijk af, door het opnieuw te starten zodra de chip meldt dat hij gereed is, zo snel mogelijk achterelkaar dus, dan haal ik een repetitiefrequentie van 10 per seconde. De tijdsduur van het afgespeelde geluid is volgens Wave Studio ook 100 ms. Continue geluid dus zonder pauzes ertussen. Met de break routine kun je een woord verkort afspelen, als ik dat doe voor dat woord dat twee adressen beslaat, dan blijkt het 200 ms stuk en het 100 ms stuk anders te klinken, maar in beide gevallen 100 ms geluid af te geven met een repetitiefrequentie van 10 per seconde precies. Meten gebeurt met een stopwatch door het 100 keer met een downcounter te programmeren.
E) Doordat de tijden afgerond worden naar veelvouden van 100 ms door de chip blijkt, dat ik bij opname van 73 fragmenten van 840 stap 10 t/m 120 ms in markeradres gevorderd ben tot hex 172, dat is dus een tijdsduur van 370*100 ms = 37 seconden rond, terwijl de som van alle opgenomen geluidsfragmenten, werkelijke tijden dus, een duur heeft van 0,5*73(120+840) = 35,04 seconden en door optelling blijkt de som van alle afgespeelde geluidsduur 34,324 s is.

Om het adagium "Hoe meer je meet hoe minder je weet" nog verder uit te diepen heb ik per woord de werkelijke tijd gemeten tussen de startpuls en het tijdstip dat de chipcorder een EOM interrupt geeft van het gestarte geluidsfragment, dat wordt gemeten per woord op de ms nauwkeurig, en die vervolgens opgeteld.
Dat geeft weer een andere tijd, hetgeen ik al verwachtte omdat de afspeeltijden niet conform de lengte van de werkelijke geluidssignalen waren die worden afgegeven. Omdat het causaliteitsbeginsel aangeeft dat de chip niet gaat spelen voor ik hem start is het waarschijnlijk dat het tijdstip dat het geluid ophoudt niet geheel overeenkomt met het EOM End-Of Messagesignaal van de 2560-chip.

Ik wilde ook nog weten uit meer praktische overwegingen wat de minimale spatie tussen twee voor opname aangeboden geluidsfragmenten is. Fragmentlengte van 500 ms daartoe geprogrammeerd met teruglopende spaties van 500 ms in stappen van 10 ms. Weer met het daarvoor iets gewijzigde C programma een assemblerlisting van een dergelijk file geproduceerd, geassembleerd en in het EEPROM van de programmer gepompt.

Resultaat: De afgespeelde fragmenten zijn volgens de ingebouwde milliseconde meting 464 ms (i.p.v. de werkelijke 500 die tussen de start- en stoppulsen ligt bij opname). De afstand tussen de markers is dan 5, d.w.z.: de volgende marker vind je door 5 bij zijn buurman op te tellen, dat komt dus wel overeen met 500 ms. En het gaat allemaal goed tot de spatie tussen twee fragmenten geslonken is tot 70 ms. Bij 70 ms gaat het voor het eerst fout, 80 ms kan dus als minimale spaties tussen twee aangeboden geluidsfragmenten worden aangehouden in de proefopzet.

Nabouw

Als je niet alleen met interesse dit verhaal gevolgd hebt maar ook wel met zoiets aan de gang wilt, bijvoorbeeld om een visueel gehandicapte amateur van dienst te zijn, en geen werk dubbel wilt doen, is source listing van de controller gratis beschikbaar. Stuur een mailtje naar mijncall@amsat.org en je krijgt hem per omgaande toegestuurd in de staat waarin hij op dat moment verkeert.
Ik ben ook bereid om als je 3 tientjes vooruitbetaalt een controller voor je te programmeren, met in het EEPROM tevens de start- en stoptijden in milliseconden van een wav-file, in machineleesbare vorm, die wav-file hoef ik er dus niet bij te hebben, alleen de start- en stoptijden. Zend een en ander aan mijncall@amsat.org.

73
PAoWV

Mogelijkheden

Als je door de bomen het bos niet meer ziet, dan hier nog een lijstje van de menukeuzes die je met de gele (loop)knop kunt kiezen:

1. Select filename
Gekozen kan dan worden met de linker rode knop uit alle tijd bestanden.

2. T-on-off
Van de in 1. gekozen filename worden dan alle aan- en uittijden in het EEPROM gemeld. Rode knop kiest de volgende.
De woorden 'on' en 'off' worden erbij vermeld, omdat dat in de praktijk wenselijk bleek. Met de gele loopknop kun je die listing weer onderbreken en terug naar het hoofdmenu.

3. Chipload
Die mode telt alle opnametijden in de gekozen tijdenfile op, zodat je de totale opnameduur kunt zien in ms. Dan heb je tevoren een indicatie of alles erop gaat.

4. Countent on/off
Deze count entries mode telt het aantal entries in de tijdentabel, dat zijn dus het aantal woordfragmenten.

5. Record
Kies je record dan moet tweemaal op de startknop gedrukt worden om het rom te gaan programmeren. Na de eerste keer volgt een waarschuwing en kun je er nog uit met de gele knop.
Na de tweede knop start hij onmiddellijk zonder debouncing delays.

6. Findadd
Find addresses bepaalt van een geprogrammeerd rom de startadressen van de geluidsfragmenten en bergt die op in EEPROM van de controller. Tevens display op de LCD van die adressen als voortgangscontrole.

7. Test
Die mode speelt het geluidsrom af van achter naar voren, zodat het ontbreken van een marker blijkt.

8. DFMA
Deze mode "display found marker addresses" geeft elke 5 seconden een nieuw adres op, zodat je dat zonder verdere bediening kunt overnemen in de assemblerlisting.

9. Piece wise replay
Geeft met de startknop van elk beginadres op het woordnummer, het begin en het eindadres en speelt het fragment af. Lijkt overbodig, is het echter niet want als er een lege marker (zonder geluid erachteraan) aanwezig is, komt die op die manier boven water.

10. Break multi entries
Deze mode speelt in EEPROM opgegeven adressen (van 07FF omlaag) per adres af, geeft de adressen op, zodat je goed kunt beoordelen waar een op te splitsen woord zijn tweede beginadres moet hebben.