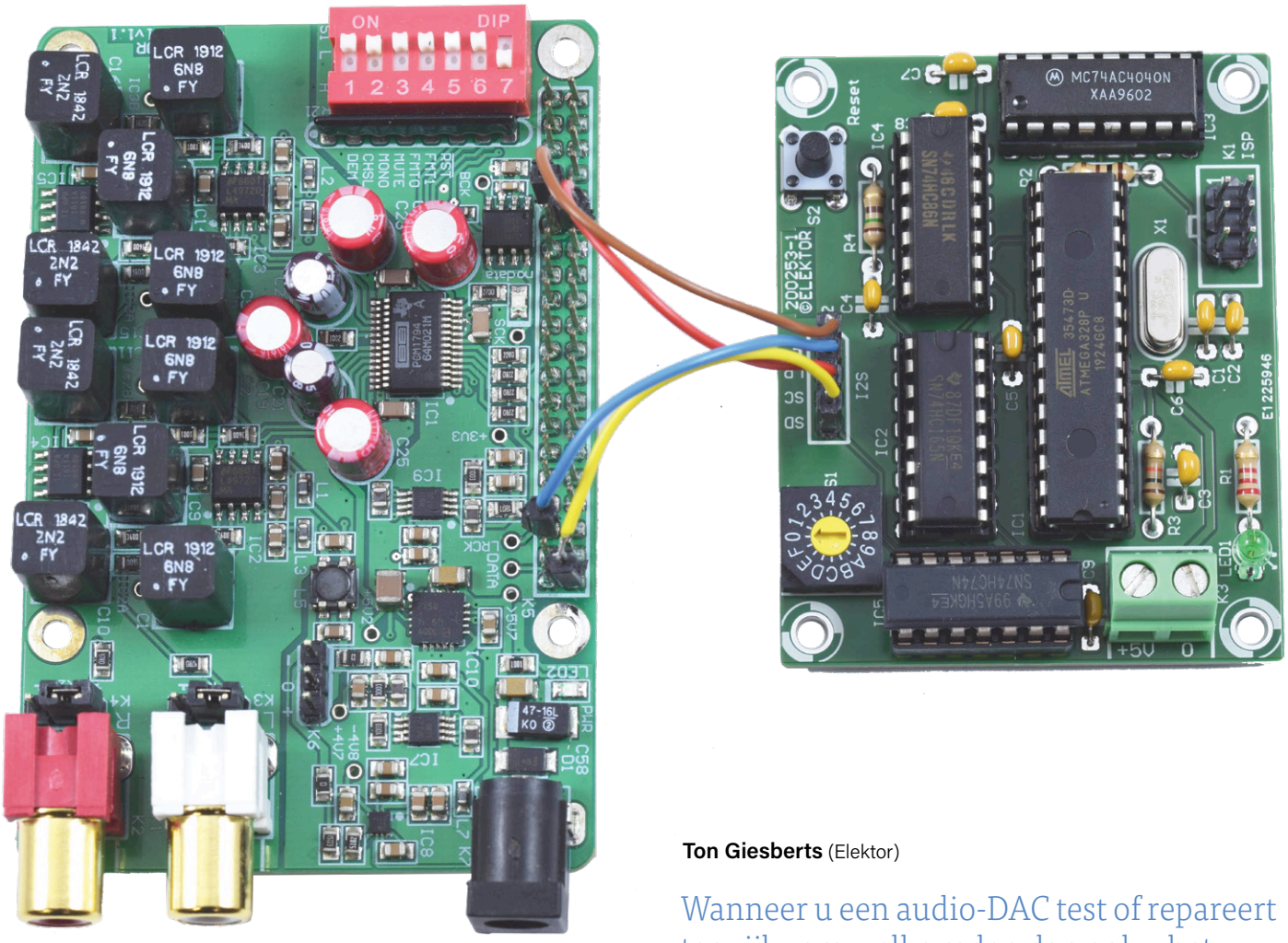
I²S testsignaalgenerator
met AVR-microcontroller

32-bit 1-kHz digitale sinus, fs 192 kHz, niveau instelbaar van 0 tot –110 dB



Ton Giesberts (Elektor)

Wanneer u een audio-DAC test of repareert terwijl – om welke reden dan ook – het analoge audio-uitgangssignaal ontbreekt of vervormd is en het onzeker is of de signaalbron een probleem heeft (software en/of hardware), dan wel het DAC-circuit zelf niet in orde is, dan kan dit project het antwoord geven. Het zeer nauwkeurige digitale sinus-testsignaal is ook ideaal om de analoge prestaties van de DAC te meten.

PROJECT-INFO

Tags
digitale audio, Raspberry Pi, DAC, I²S

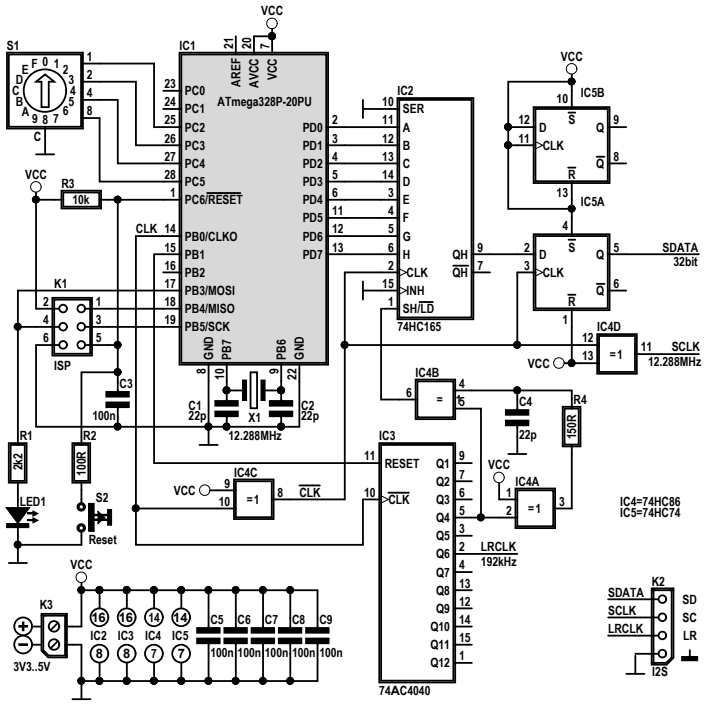
Level
beginners – gevorderden – experts

Time
ongeveer 4 uur

Tools
standaar soldeergereedschap, AVR-ISP

Cost
ca. € 15

Sinds de introductie in 1986 is de Inter-Integrated Circuit Sound (I²S) bus de 'de facto' standaard voor de overdracht van seriële digitale audiosignalen. Tijdens de ontwikkeling en test van onze 'Audio DAC voor de Raspberry Pi' [1] kwamen we op het idee om een speciale schakeling te ontwerpen die een I²S-signaal genereert om de DAC te testen zonder de RPi als signaalbron aan te sluiten. Deze schakeling kan natuurlijk ook gebruikt worden om andere audio-DAC's met I²S-ingangen te testen, op voorwaarde dat ze een bemonsteringsfrequentie van 192 kHz en 24- of 32-bits audiodata kunnen verwerken.



Figuur 1. Schema van de I²S-signaalgenerator.

Ontwerpopties

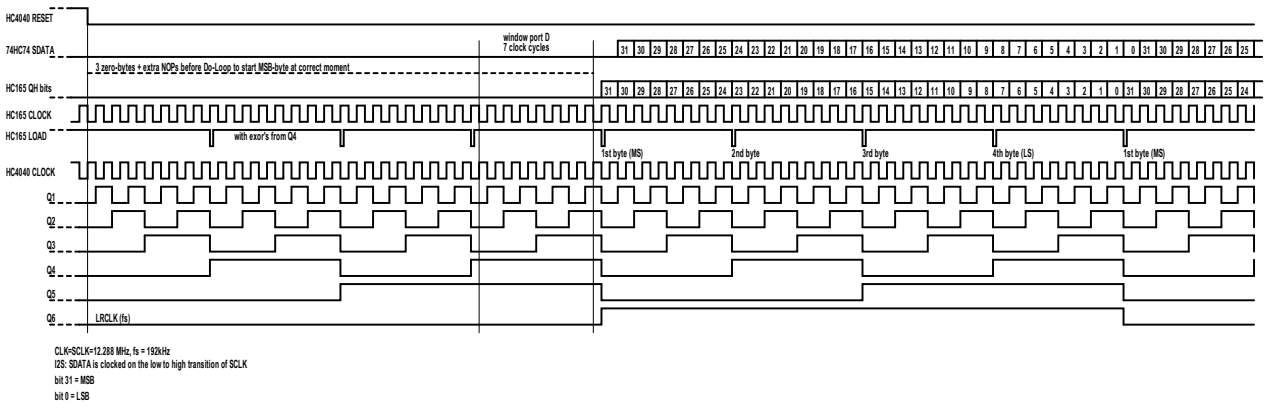
Om een I²S-testsignaalgenerator te bouwen, zou het een optie zijn om een 24-bit ADC te gebruiken met I²S-uitgangen, met een signaalgenerator (sinus) als ingang. Maar om te controleren of de analoge uitgangssignalen van de DAC inderdaad foutloos zijn, moet de sinus in het I²S-signaal perfect zijn om correcte vervormingsmetingen uit te voeren. Het testsignaal mag op geen enkele manier worden aangetast door een inferieure signaalbron of ADC.

Als alternatief kan een microcontroller worden gebruikt om het I²S-signaal te genereren, met behulp van een tabel met 32-bits monsters die nauwkeurig kan worden berekend om de kwaliteit van de audiodata te garanderen. Dit zal een signaal genereren dat perfect is voor vervormingsmetingen, in dit geval een sinus van 1 kHz met een samplefrequentie van 192 kHz.

Het ligt voor de hand om voor deze taak een microcontroller te gebruiken die I²S ondersteunt, maar waarom niet een zeer gangbare microcontroller zoals de ATmega328P? Het probleem is natuurlijk dat die I²S niet ondersteunt. Het was een hele uitdaging om een digitale sinusgenerator met I²S-uitgang te bouwen met deze microcontroller en wat extra hardware, maar dit project laat zien dat het kan! De firmware voor de ATmega is ontwikkeld in BASCOM-AVR.

Er is wat extra hardware nodig

Het doel is om een I²S-signaal te maken met 32bit-data en een samplefrequentie van 192 kHz, wat in de buurt komt van de maximale samplefrequentie van de PCM1794A die in onze RPi-audio DAC wordt gebruikt. De seriële klok (SCK of SCLK) moet 12,288 MHz zijn (2 kanalen * 192 kHz * 32 bit). Aangezien de maximale klokfrequentie van de microcontroller 20 MHz is, is de enige manier om seriële gegevens (SD of SDATA) sneller uit te voeren het gebruik van een extern parallel-in/serieel-uit schuifregister en de klok van de microcontroller te gebruiken om het schuifregister te klokken. PB0 moet worden geconfigureerd als CLKO (Clock Out) bij het



Figuur 2. Timingdiagram.

ONDERDELENLIJST

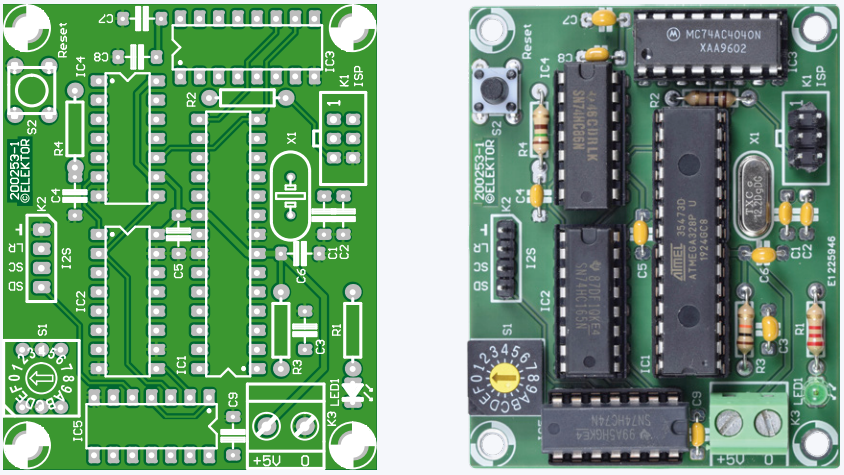
Weerstanden:
R1 = 2k2 k
R2 = 100 Ω
R3 = 10 k
R4 = 150 Ω

Condensatoren:
C1,C2,C4 = 22 p, C0G/NP0, steek 5 mm
C3,C5,C6,C7,C8,C9 = 100 n, X7R, steek 5 mm

Halfgeleiders:
LED1 = LED groen, 3 mm
IC1 = ATmega328P-PU, 20 MHz, DIP28
IC2 = 74HC165
IC3 = 74AC4040 – geen HC !!!
IC4 = 74HC86
IC5 = 74HC74

Diversen:
K1 = 2x3 pinheader, verticaal, steek 2,54 mm
K2 = 1x4 pinheader, verticaal, steek 2,54 mm
K3 = printkroonsteen 2-polig, 5,08 mm, 630 V
S1 = Rotary Coded Switch, hexadecimaal, real code, THT (bijvoorbeeld Nidec Copal Electronics SD-1010)
S2 = drukknop, 24 V, 50 mA, 6x6 mm
X1 = kristal 12,288 MHz, C-load 18 pF, 10 ppm, HC-49S

Print 200253-1 v1.1



Figuur 3. Layout van de print.

programmeren van de ATmega328-fuses. Het volledige schema van de signaalgenerator is te zien in **figuur 1**. Naast het schuifregister zijn een ripplecounter, een flipflop en enkele exclusive-OR's toegevoegd om de microcontroller te bevrijden van andere (timing-)taken dan alleen het uitvoeren van de samplebytes. Alle andere softwaretaken, zoals het berekenen van de 32-bit sinus en het aanmaken van het array van samplebytes, moeten worden uitgevoerd voordat de hoofdlus van het programma begint. De 32-bit monsters zijn verdeeld in vier bytes. Dit betekent dat de grootte van het array 4 * 192 = 368 bytes moet zijn. Het linker en rechter kanaal gebruiken hetzelfde signaal, dus elke groep van vier bytes per sample moet worden herhaald. Het aantal hiervoor benodigde programmaregels kan worden berekend:

2 kanalen × 4 bytes × 192 samples × 4 regels – 3 = 6141 programmaregels

Na de laatste samplebyte in de lus worden de drie NOP's weggelaten omdat de herstart van de lus drie klokcycli vergt, wat de '–3' in deze berekening verklaart.

De hardware nader bekeken

Het timingdiagram in **figuur 2** kan helpen bij de keuze van de externe componenten. De I²S-bus bestaat uit drie signalen: de seriële gegevens (SDATA) worden geklokt op de opgaande flank van de seriële klok (SCLK) en de wordselect-lijn (WS of LRCLK) geeft het audiokanaal aan (0 voor links, 1 voor rechts). De frequentie is gelijk aan de samplefrequentie van het digitale audiosignaal (192 kHz) en kan worden afgeleid uit de seriële klok. De seriële gegevens en de wordselect-lijn moeten beide veranderen op de neergaande flank van de seriële klok. De logische HC-familie is in principe snel genoeg om als externe componenten te worden gebruikt, hoewel de looptijdvertraging van de poorten op één punt moet worden gecorrigeerd: 12 ns vertraging is bijna 15% van de klokperiode van de microcontroller.

Een 74AC4040 (IC3, 12-traps binaire ripplecounter) wordt gebruikt om de I²S-signalen goed te timen. Deze teller telt op de neergaande flank van de klokingang. Hij heeft ook een master-reset (pin 11) die gebruikt kan worden om hem te synchroniseren met de microcontroller. De zesde flipflop (pin 2) deelt 12,288 MHz door 64 en levert exact 192 kHz, de frequentie die we nodig hebben voor de wordselect-lijn.
Aan het begin van het programma wordt de reset van de teller geactiveerd (hoog). Deze wordt net voor het begin van de hoofdlus gedeactiveerd (d.w.z. ongeveer een kwart seconde na het inschakelen, wat tijd genoeg is om de array aan te maken). Dit resetsignaal synchroniseert de gegevens aan de uitgang van het schuifregister en de wordselect-lijn. De eerste drie samples met waarde 0 en een paar NOP's worden gebruikt, zodat de eerste byte op het juiste moment aan de uitgang van SDATA staat. Er is een venster met zeven klokcycli voor het eerste MS-byte dat in het schuifregister wordt geklokt. Elk byte op poort D wordt na acht klokcycli vervangen door het volgende. Met andere woorden: het moment dat het eerste MS-byte op poort D kan veranderen ten opzichte van de laadpuls van het schuifregister kan zeven klokcycli variëren (zie timingdiagram). De gegevens aan de uitgang van poort D veranderen enkele nanoseconden na de opgaande flank van CLKO (PB0). Voor de seriële gegevens wordt een 8-bit parallel-in/serieel-uit schuifregister 74HC165 (IC2) gebruikt. Dit heeft een actief lage parallelle laadingang (pin 1) en een klok (pin 2) met inhibit (pin 15, actief laag) die dezelfde functionaliteit hebben (beide intern verbonden met een OR-poort). Afhankelijk van de plaatsing van de componenten kan het verwisselen van de verbindingen de routing vereenvoudigen. De gegevens worden op de opgaande flank van de klok verschoven. De seriële ingang (pin 10) wordt niet gebruikt en met massa verbonden. De laadpuls (LD, actief laag) voor het schuifregister wordt afgeleid van de telleruitgang Q4 met behulp van IC4 (quad 2-Input exclusive-OR 74HC86). Het signaal naar pen 4 van IC4B wordt geïnverteerd en vertraagd door de looptijdvertraging van IC4A en een extra vertraging van enkele nanoseconden door R4/C4. Door de exclusive-OR functie geeft elke verandering van Q4 een korte actieve lage puls aan de uitgang van IC4B. De puls is lang genoeg om de nieuwe gegevens in het schuifregister te laden, maar kort genoeg om inactief te zijn voor de opgaande flank van de klok. Om de gegevens van poort D op het juiste moment

in het schuifregister te laden moet de puls net na de opgaande flank van de klok actief zijn (pen 1). Dit betekent dat de klok van het schuifregister moet worden geïnverteerd, wat door IC4C wordt gedaan.
Het meest significante bit van de seriële data van de I²S-bus bevindt zich één klokperiode na de wijziging van wordselect. Een extra D-type flipflop is nodig. We hebben gekozen voor een 74HC74 (IC5), een dubbele D-type flip-flop met set en reset en positive edge-trigger. Hierdoor wordt het signaal voor SDATA vertraagd. Om dit te compenseren wordt het geïnverteerde kloksignaal van IC4C weer geïnverteerd door IC4D om de seriële klok te synchroniseren met de seriële gegevens.

Instellen van het uitgangsniveau

In plaats van alleen een sinusgolf van 1 kHz met een vast niveau te produceren, worden vier ingangen van poort C (met interne pullups) aangesloten op een hexadecimaal gecodeerde draaischakelaar (SD-1010) om het uitgangsniveau aan te passen. Deze functie kan worden gebruikt om de lineariteit van de DAC te controleren. In plaats daarvan had een vierpolige DIP-schakelaar kunnen worden gebruikt, maar het wijzigen van niveaus is eenvoudiger met de draaischakelaar, net als bij een analoge volumeregelaar.
Er zijn draaischakelaars van het type 'echte code' en van het type 'complementaire code.' In dit project gebruikten we de eerste versie, maar andere types kunnen worden gebruikt als de software wordt gewijzigd. In positie '0' staan de vier schakelaars open. Om het uitgangsniveau voor het digitale audiosignaal te wijzigen, moet de microcontroller worden gereset om alle waarden in de sample-array opnieuw te berekenen, vandaar de aanwezigheid van knop S2 die moet worden ingedrukt om de microcontroller te resetten als het uitgangsniveau wordt gewijzigd. Een Select-Case instructie wordt gebruikt om de schaalfactor U (type Double) op de juiste waarde in te stellen voor elke niveau-instelling.
De juiste waarden voor de schaalfactoren voor elk uitgangsniveau worden in het programma gedefinieerd als constanten, niet alleen om extra berekeningen te voorkomen, maar ook omdat berekeningen met de LOG-functie van BASCOM in het programma niet nauwkeurig genoeg zijn.

Berekening van de sinus

Zoals eerder vermeld, berekent het programma de sinustabel onmiddellijk na het inschakelen of resetten, rekening houdend met de niveau-instelling van draaischakelaar S1. Het doel van dit project was niet alleen om 'zomaar' een I²S-signaalgenerator te bouwen, maar ook om een perfect 1 kHz sinus-testsignaal met 32-bit nauwkeurigheid te genereren. Aanvankelijk werd de BASCOM-AVR instructie voor SIN(x) gebruikt in de berekeningen, maar dat is niet nauwkeurig genoeg, zoals de volgende voorbeelden laten zien:

```
DIM Pi,A,X As Single
Pi = 3.14159265358979323846264433
X = Pi / 2
A = SIN(X)
Print "Sin(Pi/2) = " ; A
```

Dit stukje code resulteert in: sin(pi/2) = 0,99999332, terwijl het resultaat precies 1 moet zijn. En voor X = pi/6 is de uitkomst 0,499993796, maar zou het precies 0,5 moeten zijn. Voor het berekenen van een extreem nauwkeurige sinus is dit dus gewoon niet goed genoeg. De enige optie is om de sinus te berekenen met behulp van het Taylorpolynoom voor SIN(X) met voldoende termen:

SIN(X) = X-(X^3)/3!+(X^5)/5!-
(X^7)/7!+(X^9)/9!-(X^11)/11!+
(X^13)/13!-(X^15)/15!

In BASCOM-AVR kunnen berekeningen slechts op twee operanden worden uitgevoerd; daarom wordt het polynoom berekend in veel coderegels, zoals te zien is in de broncode van dit project.
Om de zaken te versnellen worden de faculteiten (3!, 5! enzovoort) niet berekend maar worden in plaats daarvan constanten gebruikt (3! = 6 … 15! = 1307674368000). Alle variabelen in de berekeningen zijn van het type double, 64-bit signed binair (8 bytes, 5 x 10^–324 tot 3,4 x 10^308). Met het Taylorpolynoom krijgen we de volgende, nauwkeurigere resultaten:

sin(pi/6) = 500E–3
sin(pi/2) = 999,999999993977E–3

Om het resultaat van de berekening van SIN(X) op te splitsen in vier bytes moet het eerst worden omgezet in een variabele van het type long, 32-bit signed binair (–2147483648 tot 2147483647). Het resultaat van de berekening van SIN(X) ligt in het bereik van –1 tot 1, als we willen dat de 32-bit gegevens een full-scale niveau hebben moet de waarde van SIN(X) vermenigvuldigd worden met (2^32)/2 – 1:

SINX = SINX * U
met U = 2147483647

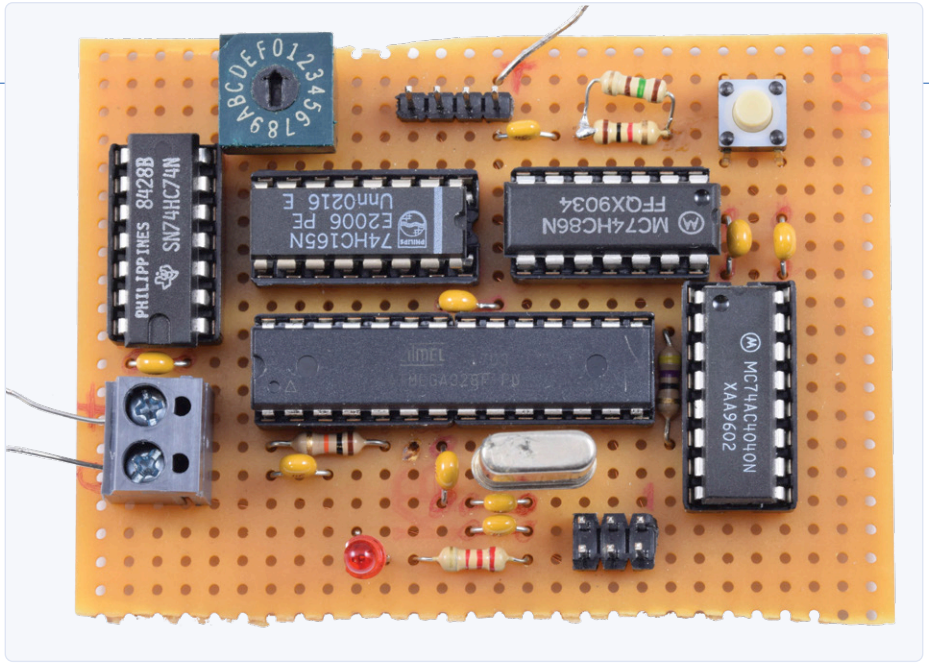
De conversie naar long kan worden gedaan in slechts één opdracht, een long-variabele krijgt de waarde van een double-variabele. Nu is het resultaat van de berekening beschikbaar als een signed 32-bit waarde:

SINXlong = SINX

Het splitsen van deze waarde in vier bytes is slechts een kwestie van het verschuiven van de bits van SINXlong en het opslaan ervan in vier variabelen van het type byte. De subroutine met de naam SampleX doet al deze berekeningen voor elke waarde van X. Maar de berekening is alleen echt nauwkeurig voor X = –π/2 tot +π/2. Dus eerst wordt de sinus berekend van –π/2 tot +π/2, waarbij 97 monsters worden genomen (vier bytes per monster). De rest van de array wordt aangevuld door de elementen van de eerste array te spiegelen per groep van vier bytes, waarbij 95 samples worden genomen. Sample 193 is hetzelfde als de eerste van het array (vier bytes). De Do-Loop start daar opnieuw.
Het zou mogelijk zijn geweest om een array te maken met alleen samples van de sinus van –π/2 tot +π/2 en de juiste array-elementen in de hoofdlus te selecteren om de gegevens

WEBLINKS

[1] **Audio DAC for Raspberry Pi**: www.elektormagazine.com/labs/audio-dac-for-rpi-networked-audio-player-using-volumio
[2] **Download Gerber-bestanden en software**: www.elektormagazine.nl/200253-04
[3] **Elektor Labs-pagina van dit project**: www.elektormagazine.com/labs/32-bit-i2s-sine-wave-generator-200253



Figuur 4. Het prototype van de generator op gaatjesprint.

voor een volledige sinusperiode bij poort D te completeren. Maar in plaats van een sinus kan een complexer signaal dat niet symmetrisch is, interessant zijn voor andere doeleinden. En als een volledige array wordt gebruikt, is de hoofdlus beter leesbaar.

Met behulp van het schuifregister heeft de microcontroller acht klokcycli om elk byte te verwerken.

In BASCOM-AVR heeft de instructie `PORTD = A(n)`, waarbij A(n) een element van een byte-array is, bij gebruik in de Do-Loop slechts vijf klokcycli nodig, verrassend snel. Om elk byte acht klokcycli te laten duren moeten na elke byte drie NOP's worden toegevoegd. Het duurt drie klokcycli om de Do-Loop opnieuw te starten, wat perfect is om een volledige periode van 192 samples van een 1kHz-sinus te produceren. Dus het enige wat de hoofdlus van het programma hoeft te doen is de bytes van de array uit te voeren naar poort D.

Bouw van de hardware

Voor dit project hebben we de in **figuur 3** getoonde print ontworpen, de layout en de Gerber-bestanden zijn als download beschikbaar. U kunt ook een stuk gaatjesprint gebruiken, zoals onze ontwerper in de eerste prototypefase heeft gedaan (**figuur 4**).

Als u de 74**AC**4040 nergens kunt vinden, kunt u de verbinding tussen pin 13 en 14 van IC4 doorsnijden. Door pen 13 met een jumper aan massa te leggen *kan* een 74**HC**4040 worden gebruikt, maar de timing is verre van optimaal en kan een probleem vormen voor de te testen DAC. Deze correctie werkt wel met onze RPi DAC. De draden van K2 van de generator naar de DAC moeten zo kort mogelijk worden gehouden.

De voedingsspanning van deze schakeling moet overeenkomen met de spanning van de geteste DAC. Bij V_{CC} = 3,3 V is het stroomverbruik iets meer dan 20 mA.

Software wijzigen...

Het programma gebruikt 77% van het flashgeheugen. Alleen al het berekenen van de sinus zoals hierboven beschreven heeft ongeveer 10% nodig. Er is dus nog wat ruimte over in het programmageheugen om de code uit te breiden en extra functies toe te voegen. Merk op dat de gratis demoversie van BASCOM-AVR niet kan worden gebruikt om de software te compileren vanwege de beperkingen van het programmageheugen, u zult een geregistreerde versie moeten kopen als u het programma wilt wijzigen.

De software-download voor dit project [2] bevat het HEX-bestand van de originele firmware, dus dit project kan worden gebouwd zonder een gelicentieerde BASCOM-AVR-versie; met slechts een AVR-ISP programmeerinterface en programmeersoftware zoals Atmel AVR Studio of AVRDUDE kunt u uw ATmega128 aan de praat krijgen.

...of hardware

Er zijn nog steeds enkele I/O-pennen van de microcontroller ongebruikt die kunnen worden gebruikt om extra functies aan te sturen, zoals het wijzigen van de golfvorm of de frequentie. Natuurlijk moet de software worden aangepast om dergelijke functies te ondersteunen. Microcontrollerpennen PB2, PC0 en PC1 kunnen zonder probleem gebruikt worden, terwijl PB4 en PB5 speciale pinnen zijn voor In System Programming (ISP); er moet op gelet worden dat extra aangesloten componenten deze programmeerinterface niet verstoren.

LED1 is toegevoegd als indicatie dat de processor bezig is met het berekenen van de audiosampledata, maar aangezien dit in slechts een kwart van een seconde klaar is, kan PB3 ook worden gebruikt voor extra functies. In dat geval kan R1 worden aangesloten op de voeding als aan/uit-indicatie.

Als u functionaliteit toevoegt aan deze I²S-signaalgenerator, of als u nog andere opmerkingen of suggesties heeft met betrekking tot dit project, vergeet dan niet deze met ons en andere lezers te delen op de Elektor Labs-pagina [3]!

200253-04

Vragen of opmerkingen?
Hebt u vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur of naar de redactie van Elektor, via redactie@elektor.com.

Een bijdrage van
Idee, ontwerp en tekst: **Ton Giesberts**
Schema en illustraties: **Ton Giesberts, Patrick Wielders**
Redactie: **Luc Lemmens**
Vertaling: **Jelle Aarnoudse**
Layout: **Giel Dols**