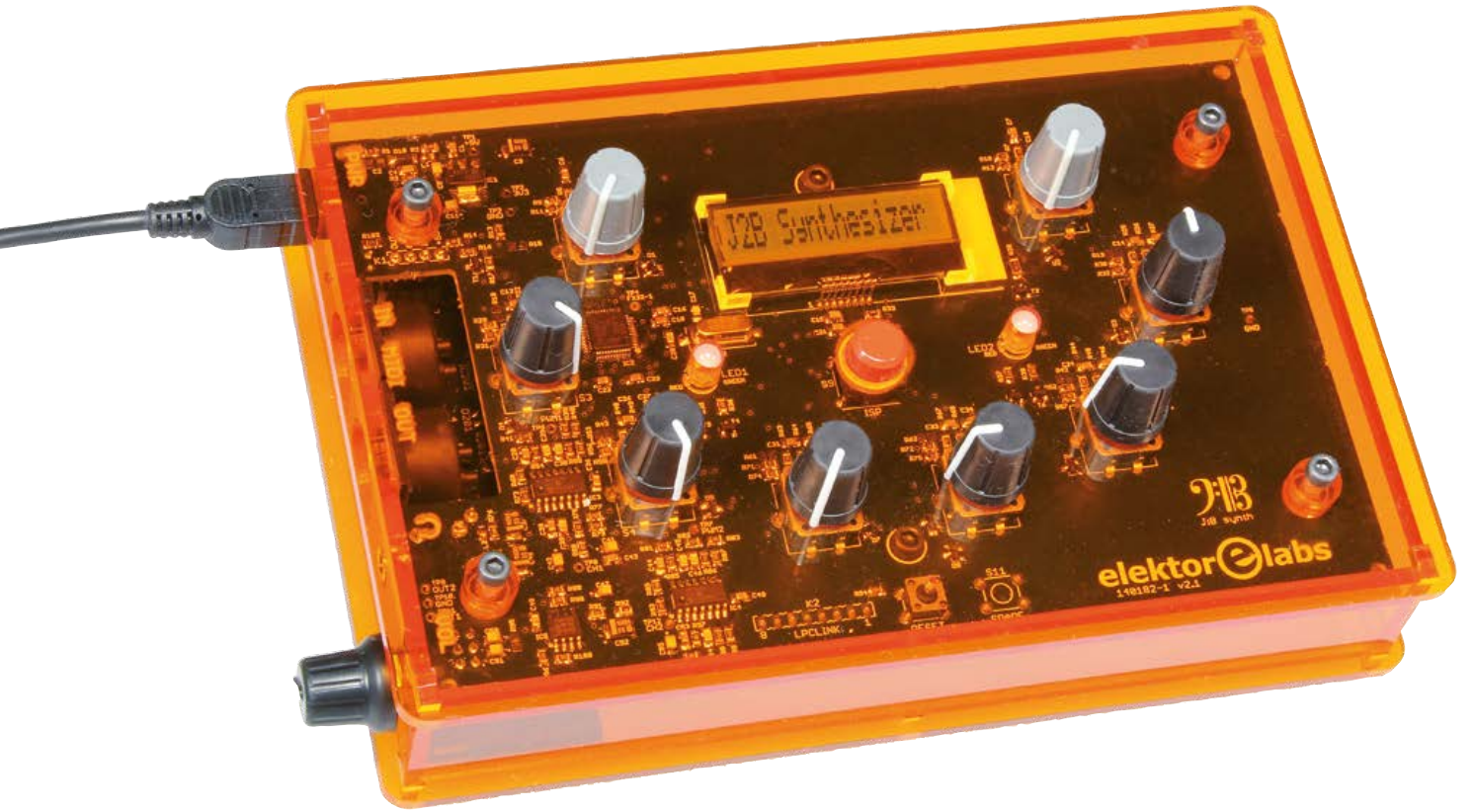
J²B Synthesizer

Een open platform voor digitale muziek

Clemens Valens
(Elektor-lab)



Dit project is ontstaan nadat ik de Atmegatron muzieksynthesizer van Soulsby Synthesizers [1] had ontdekt. De Atmegatron is gebouwd rond een ATmega328 8-bits AVR microcontroller van Atmel, dezelfde µC als in de Arduino Uno. De Atmegatron heeft een paar intrigerende ontwerp-eigenschappen die mijn aandacht trokken. Die leidden uiteindelijk tot het project in dit artikel.

Technische eigenschappen

- Monofone 9-bits synthesizer
- 32 golfvormen + vrij definieerbare
- 15 filtertypes
- 2 omhullende-generatoren
- LFO met 16 golfvormen
- Arpeggiator voor 15 patronen
- 16 patch-geheugens
- 6 live controls
- MIDI
- Patch opslaan en laden via MIDI
- NXP LPC1347 32-bits ARM Cortex-M3 microcontroller
- 2 uitgangskanalen
- Open-source- & open-hardware-ontwerp

Voordat we dit project induiken, wil ik graag beginnen met een waarschuwing aan de lezer die vooral geïnteresseerd is in de theoretische achtergronden van klanksynthese op de Atmegatron. In dit artikel zet ik uiteen hoe ik de Atmegatron heb overgezet (ge-*port*) naar mijn eigen hardware-platform. Hoe de klankgenerator van de Atmegatron werkt, wordt in dit artikel maar heel beknopt beschreven. Wilt u zich daarin verdiepen, dan raad ik aan om figuur 2 te bekijken en de source-code [2] te bestuderen in plaats van dit artikel.

Atmegatron

Ah, u bent er nog. Alle taken die er zijn in de Atmegatron neemt de microcontroller voor zijn rekening, van het genereren van klank tot de interactie met de gebruiker. Communicatie met andere instrumenten is ook mogelijk via een MIDI-poort (*Musical Instrument Digital Interface*). Paul Soulsby heeft dit alles onder weten te brengen in het programmageheugen van de ATmega, slechts 32 KB! Dat vind ik knap, zeker als je in aanmerking neemt dat de software een Arduino-sketch in C/C++ is, wat nogal wat overhead introduceert. De software is vrij beschikbaar onder een open-source-licentie. U kunt deze gratis downloaden. Het hardware-ontwerp van de Atmegatron is helaas niet 'open', maar kunnen we wel reconstrueren door de software te bestuderen.

De Atmegatron is bedoeld voor live-optredens en is daarom voorzien van zes potmeters waarmee geluidsparameters direct te wijzigen zijn. Samen met een functie-keuze en een encoder voor de functie-waardes heeft een Atmegatron dus acht knoppen. Verder is er nog een drukknop die tussen twee modi schakelt, groen of rood. Met LED's wordt de toestand van het apparaat aangegeven, er is geen alfanumeriek display.

Synths van gevestigde fabrikanten produceren veelal wat gelikte, zoetgevooisde klanken. Daar moet u de Atmegatron niet mee vergelijken. Een aantal algoritmes op de Atmegatron produceert vervormde, wat ruigere klanken. Hiermee onderscheidt hij zich duidelijk van de grote jongens.

De synthesizer produceert klank door middel van de PWM-generatoren in de µC. De externe digitaal/analoog-converter doet uitsluitend dienst als anti-aliasing-filter.

Een open-source klankgenerator met acht draaiknoppen (zes potmeters en twee draai-encoders) die in 32 KB past, zou je die kunnen *porten* naar Elektors J²B-platform dat we presenteerden in 2011 [3]? J²B is ontworpen rond de LPC1343, een 32-bits ARM Cortex-M3 CPU van NXP. Net als de ATmega328 heeft hij 32 KB programmageheugen en alle randapparatuur voor de synthesizer aan boord, inclusief de nodige PWM-faciliteiten. Verder kunnen er maximaal negen draai-encoders op het J²B-board, of acht en een druktoets - perfect! Een EEPROM voor de klank-presets is er nog niet, maar daar verzinnen we wel wat op.

Zoals gezegd heeft de Atmegatron geen LCD, visuele informatie over zijn toestand is te zien aan LED's. Het J²B-board heeft wel een LCD, er is zelfs keuze uit drie afmetingen.

Wat stond me te doen? Drie hoofdtaken:

1. de Arduino sketch voor de ATmega328 porten naar een Eclipse/LPCXpresso-project voor de LPC1343;

Figuur 1.
Dit project is ontstaan uit de Atmegatron.

2. zes analoge potmeters vervangen door zes digitale draai-encoders;
3. de tweekleuren-LED's vervangen door een LCD.

Taak 1

Microcontroller-code porten kan een meer of minder ontmoedigende opgave zijn, al naar gelang de kwaliteit van de code. Een goed gestructureerd project is veel makkelijker over te zetten dan spaghetti-code. Nog een bepalende factor is het abstractieniveau van de hardware: code die registers en randapparaten aanstuurt middels functie-aanroepen is veel makkelijker te porten dan code die rechtstreeks de hardware aanspreekt als het zo uitkomt. Verder heb je al die functies liefst bij elkaar in één file, en niet verspreid over allerlei bestandjes. Nu is de Atmegatron-code goed gestructureerd en gedocumenteerd, dus prima porteerbaar. Ik heb er een flink deel van gedaan terwijl ik intussen op tv naar een film met Bruce Willis zat te kijken. Het meeste werk was het maken van header-files voor het Eclipse/LPCXpresso-C-project (voor Arduino-sketches zijn die headers niet nodig). Valkuilen die je hier vaak tegenkomt zijn inconsequent gebruik van datatypes in de software zelf en incompatibiliteit van datatypes tussen compilers (van AVR GCC voor Arduino naar ARM GCC voor Eclipse/LPCXpresso). Beide problemen geven vergelijkbare bugs:

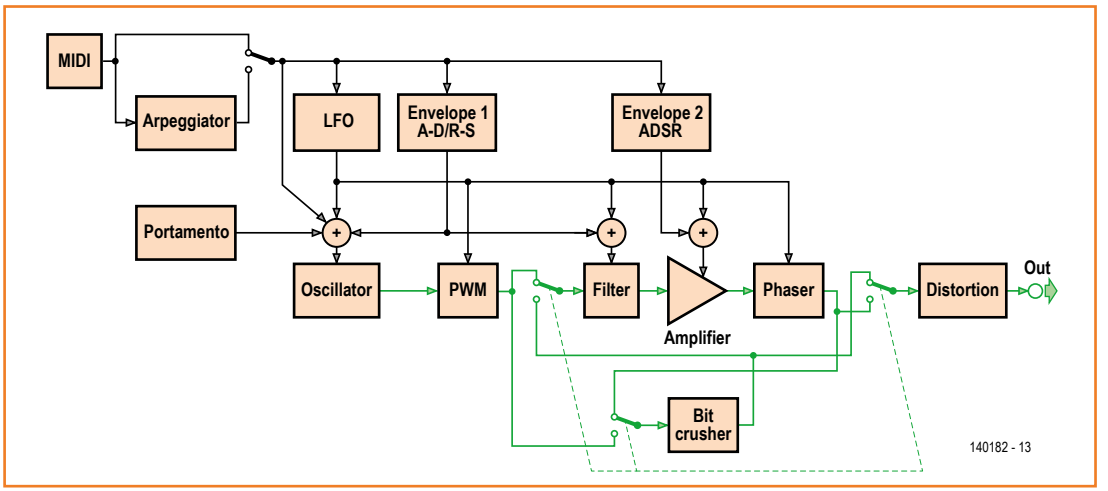
data-overflow (variabelen passen niet meer in hun gereserveerde geheugenruimte) en onbedoelde tekenverwisselingen (negatieve getallen die als positief worden opgevat en omgekeerd). Dit soort problemen is te vermijden door consequent gebruik van goed gedefinieerde datatypes met een eenduidig bereiken en teken. Zo is het beter om `uint8_t` te gebruiken voor een 8-bits getal zonder teken dan `unsigned char`. Evenzo is `int16_t` of `int32_t` beter dan `int`, want de precieze grootte van een `int` is sterk platform-afhankelijk. Nog beter is als het soort data blijkt uit de data-typering. Maak bijvoorbeeld een datatype `sample_t` voor bemonsterde waardes en houd daar consequent aan vast in alle code.

Porten wordt moeilijker naarmate hardware-platforms meer verschillend zijn. Timers zijn meestal geen probleem, want die functioneren bij elke µC wel ongeveer hetzelfde. Je moet alleen wel goed begrijpen hoe een bepaalde timer precies wordt geacht te werken. Maar stel dat een functie op het bronplatform er op het doelplatform gewoon niet is, wat dan? Of als randapparatuur wel vergelijkbaar is, maar zich toch net iets anders gedraagt? Tegen dat probleem liep ik op met de PWM-module op de LPC1343, die iets anders bleek te werken dan die op de ATmega328. Er kwam wel geluid uit, maar met vervorming.

Ik zal het uitleggen. De PWM-module op AVR-µC's vergelijkt continu een teller met een bepaalde drempel. De PWM-uitgang is de resultante van die vergelijking; die bepaalt de duty-cycle van het PWM-signaal. Tellerwaarde groter of gelijk aan drempel geeft PWM-uitgang laag. Tellerwaarde kleiner dan drempel geeft PWM-uitgang hoog. Is de PWM-uitgang hoog en daalt de drempelwaarde tot onder de tellerwaarde, dan wordt de PWM-uitgang onmiddellijk laag. In C-achtige pseudocode:

```
counter = counter + 1
if (counter == max) then counter
= 0
if (counter >= threshold) then
PWM = 0
else PWM = 1
```

Dit proces is bijna identiek op de LPC1343. Bijna, maar niet helemaal. De PWM-uitgang is niet het resultaat van een continue ver-



Figuur 2.
Blokschema van de klankgenerator van de synthesizer. De blokken PWM, bit crusher en distortion geven originele, ruige klanken.

gelijking zoals hierboven, maar verandert alleen op het moment dat teller en drempel precies aan elkaar gelijk zijn. Dit heet een match. Wordt in dat geval de drempelwaarde iets lager dan de tellerwaarde, dan komt de match niet onmiddellijk, maar pas als de teller nog bijna een heel rondje heeft geteld, naar zijn maximum toe en dan vanaf nul weer naar de nieuwe drempel. In pseudocode:

```
counter = counter + 1
if (counter == max) then
  {
    counter = 0
    PWM = 1
  }
if (counter == threshold) then PWM = 0
```

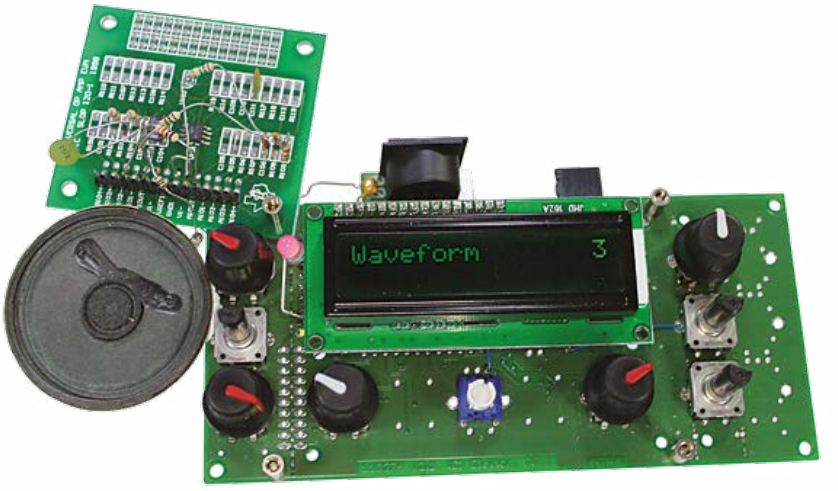
Voor de klankgenerator maakt dit hoorbaar verschil. Laten we die dus maar eens wat beter bekijken.

De Atmegatron klankgenerator

De klankgenerator (**figuur 2**) is verantwoordelijk voor het creëren van de klanken die de synthesizer produceert. In ons geval berekent hij de klank per blok van 32 samples. Zo'n blok is altijd één periode van de golfvorm (d.w.z. de samplefrequentie is niet constant). De berekeningen beginnen met een voorgedefinieerde golftabel van 32 samples voor één periode (er zijn 32 van die tabellen, maar je kunt ook je eigen golfvormen maken). De samples worden gefilterd, op diverse manieren bewerkt en dan in een uitgangsbuffer opgeslagen. Die uitgangsbuffer wordt steeds zo snel mogelijk bijgewerkt in de hoofdlus op de Arduino. Dit gedraagt zich dus eigenlijk als een taak op de achtergrond, waaraan geen prioriteit is toegekend; het wordt uitgevoerd als het kan. Dynamische parameters, zoals omhullende en modulatie, worden gesynchroniseerd met een milliseconde-timer en zijn daardoor afhankelijk van de uitvoeringssnelheid van de hoofdlus (die hoger is).

De toonhoogte van de klank wordt bepaald door een timer die 32 keer sneller loopt dan de toonhoogte, om te compenseren voor de omvang van de golftabel. Bereikt de timer zijn eindwaarde (afhankelijk van de toonhoogte), dan wordt de volgende sample uit de uitgangsbuffer genomen en wordt de PWM-drempel (de duty-cycle) ingesteld volgens die nieuwe sample-waarde. Dankzij die continue vergelijking gebeurt dit op de AVR zonder merkbare vertraging. Op de LPC1343 is die vertraging wél merkbaar. Ligt de nieuwe drempel vlak onder de vorige, dan wacht je nagenoeg een hele PWM-periode op een match. Die wachttijd is waarneembaar als een zacht ritmisch geklik. En dat is ongewenst.

Dit heb ik opgelost door de PWM-update op de LPC op interrupt-basis te laten werken. Het PWM-signaal heeft een frequentie van 140,625 kHz, maar de µC kan interrupts op die snelheid nog probleemloos afhandelen. De interrupt-routine voor de toonhoogte-timer werkt nog steeds de uitgangsbuffer af, maar ik update de PMW-drempel niet meer direct met een nieuw sample, maar met een variabele als tijdelijke opslag. De PWM-interrupt-routine leest deze variabele en gebruikt die om de duty-cycle te actualiseren. Op deze



Figuur 3.
Het prototype op de J²B met acht draai-encoders en een 2x16 LCD.

manier wordt de PWM-update netjes gesynchroniseerd met het PWM-signaal zelf. Het klikken is verdwenen.

Geheugentekort

Toen mijn porteerproject bijna voltooid was, liep ik tegen een nieuw probleem aan: niet genoeg geheugen. Dat had ik niet verwacht, want de AVR en de LPC hebben evenveel programmageheugen (32 KB) en ik zou denken dat de LPC-code efficienter met geheugen zou omspringen, want het wordt gecompileerd als zogeheten *thumb code*, wat bedoeld is om programmaruimte te besparen. Is de Arduino-compiler echt heel goed? Of gaat de AVR heel efficiënt om met de code?
Hoe dan ook, de toename kwam deels door de hardware-abstraction-library van de LPCXpresso en deels door de newlib-library die standaard in een LPCXpresso-project wordt meegelinkt. Die bevat niet alleen de standaard C-library, maar ook printf-achtige debugging, zelfs in release-mode. Dat kan eruit als je het niet nodig hebt. Deze optie was niet snel te vinden, maar het was de moeite van het naspeuren wel waard. In de project-eigenschappen opent u 'C/C++ build' en dan 'Settings'. Vervolgens krijgt u via 'Manager Linker Script' in de lijst onder 'Tool Settings' toegang tot de library. (Mogelijk moet u eerst de 'MCU Linker' uitklappen om hierbij te komen).
Met enig experimenteren had ik met de instelling 'Redlib (none)' resultaat, met elke andere library op '(none)' trouwens ook. Hiermee kwam er een flink stuk geheugen vrij. Ik moest alleen nog wel de ondersteuning voor de EEPROM maken en een gedeelte van de gebruikersinterface. Maar goed, taak 1 was hiermee afgerond.

Taak 2

Potmeters vervangen door draai-encoders lijkt misschien triviaal, maar is het beslist niet. Een snelle draai van minimaal naar maximaal voelt met een draai-encoder heel anders dan met een potmeter. Met een draai-encoder kun je een parameter weer heel precies instellen, maar dan zit je je suf te draaien als je van het ene uiteinde van het bereik naar het andere wilt - tenzij je een truc toepast. We willen iets maken dat net zo aanvoelt als een potmeter, maar met behoud van de superieure precisie van de draai-encoder. Dat wordt hoekversnelling meten, of iets dergelijks. Nu ben ik meestal meer het type dat eerst uitprobeert en dan gaat meten, maar dit keer leek het me toch beter om het andersom te doen. Ik heb mijn favoriete encoder (24 pulsen per omwenteling) aan een oscilloscoop gehangen en ben draaisnelheden gaan meten. Met een snelle draai aan de encoder bleek ik pulssnelheden tot wel 100 Hz te kunnen halen. Nu wilde ik van 0 naar 255 kunnen gaan met één snelle draai en niet in tien hele omwentelingen. Toen ik eenmaal wist welke pulssnelheden ik kon verwachten, heb ik een algoritme ontworpen dat de pulssnelheid meet en daar een versnellingsfactor uit berekent. Het resultaat mag er wezen, al zeg ik het zelf. De knoppen laten zich zowel snel als langzaam makkelijk en intuïtief bedienen.

Taak 3

De derde taak was het vervangen van de tweekleuren-LED's door een alfanumeriek LCD. Om de LED's te handhaven had ik de µC moeten voorzien van port expanders en het plan was nu juist om zo dicht mogelijk bij het J²B-board te blij-

ven. Het voordeel van een LCD is dat je er meer informatie op kunt laten zien - veel gebruiksvriendelijker, geen handleiding nodig. Met acht encoders op het J²B-board ligt een 2x16 LCD voor de hand, anders past het niet allemaal op de print (zie **figuur 3**). Goede display-teksten bedenk je niet zomaar even, maar nog moeilijker vond ik het om een goede weergavemethode voor de stand van de knoppen te bedenken. De makkelijke manier is natuurlijk om twee rijen van drie waardes te laten zien, maar dan is het niet zo duidelijk welke waarde bij welke knop hoort. Na een heleboel denkwerk vond ik toch een aardige oplossing (vind ik), in de vorm van schuif-icoontjes. Op een standaard LCD kun je maximaal acht karakters van 5x7 zelf definiëren. Dat is net genoeg om een verticale schuif met zeven posities te maken. Verdeeld over twee rijen met twee karakters boven elkaar heb je 14 posities. Met nog twee speciale karakters voor 0 en Maximum heb je dan een schuif met 16 posities. Een speciaal algoritme detecteert of de gebruiker een functie/waarde-knop bedient of een 'live' klankparameter. Daardoor kan de software automatisch schakelen tussen twee pagina's en krijgt de gebruiker de juiste informatie te zien op het juiste moment.
De rode en groene modi van de Atmegatron heb ik uitgevoerd met een tweekleuren-LED, hoewel ik daar eigenlijk liever het backlight van de LCD voor had gebruikt.

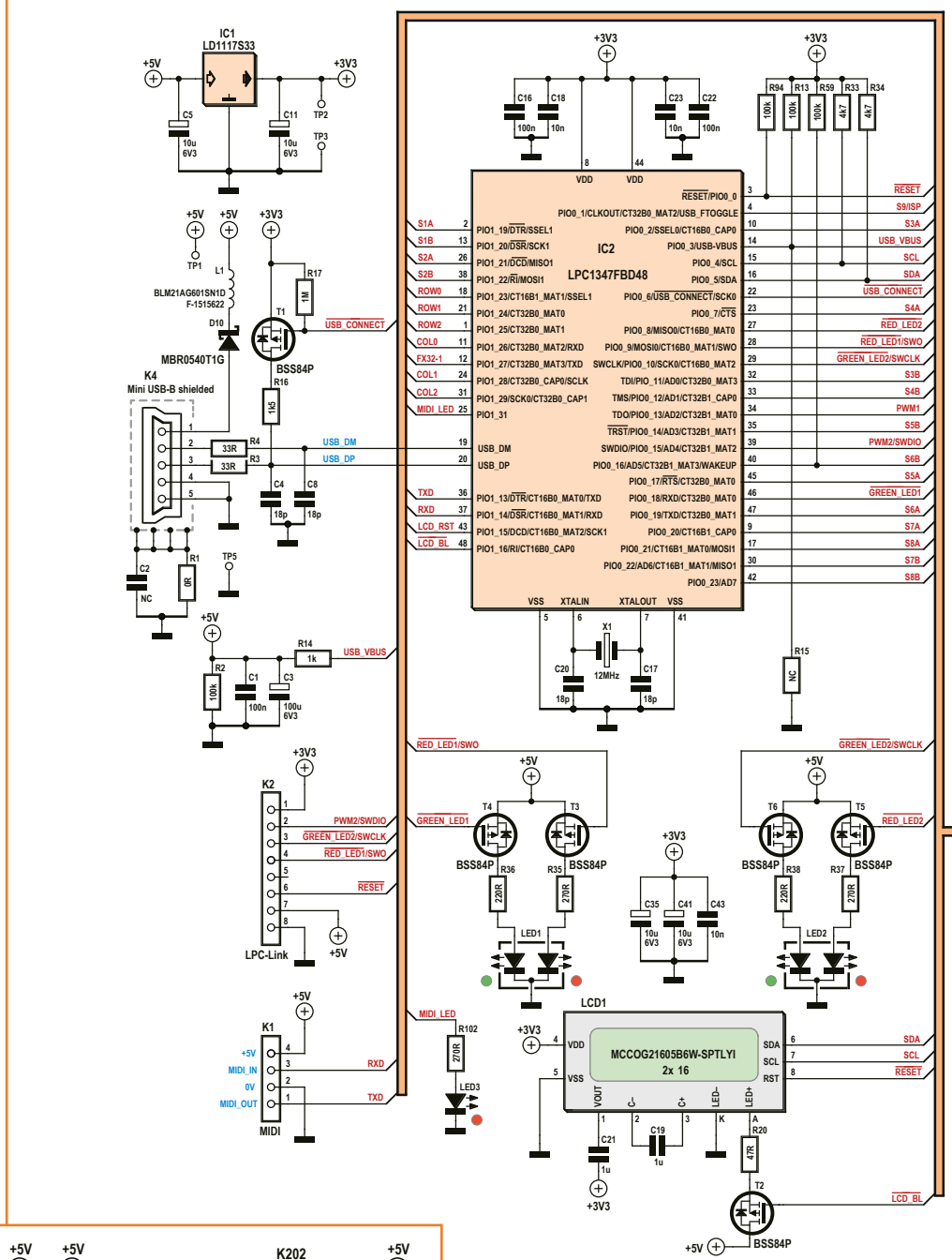
Over de geluidskwaliteit

De Atmegatron is ontworpen als een low-fi 8-bits synthesizer. Hij heeft niet de pretentie van een high-end muziek-synthesizer en zo klinkt-ie dan ook niet. Dankzij een speciale distortion-module is hij goed in agressieve, ruigere klanken en computerachtige piepjes. Stelt u zich een jaren-80 Casio voor op anabole steroïden. Niettemin zijn er uitstekende klanken mee te maken en hij is vooral ook leuk om te bedienen. De arpeggiator is een handige aanvulling. Ik heb de geluidskwaliteit een beetje verbeterd met de J²B-implementatie, want de LPC1347 beschikt over 16-bits PWM terwijl dat bij de Atmegatron maar 8-bits is (16-bits PWM is op een AVR wel mogelijk, maar te traag voor deze toepassing). Daardoor kon ik de PWM-diepte verhogen met een bit, dus het is nu een 9-bits synthesizer. De PWM-frequentie is ook meer dan verdubbeld, waardoor anti-aliasing-filtering verbeterd is (5de orde, was 3de orde), wat ook weer de geluidskwaliteit nog iets verbetert.
Ruimte voor verbetering is er echter te over. Zo hebben de algoritmes van de klankgenerator de neiging om hun uitgangen af te kappen tot 8 bits, wat natuurlijk een beetje zonde is bij een 32-bits µC. De golftabellen zouden langer kunnen. Filteren gaat nu met *floating point* berekeningen. Dat werkt prima, maar het kost wel tijd en geheugen. Als je zou filteren met vaste-komma-berekeningen maak je rekencapaciteit vrij voor andere algoritmes. Je zou er ook een virtuele synthesizer van kunnen maken via de USB-poort en/of via MIDI over USB.
De J²B-synthesizer is een betaalbaar platform waarmee je kunt experimenteren met klanksynthese op de computer, rechtstreeks op het instrument. Prog & Play, noemen we dat.

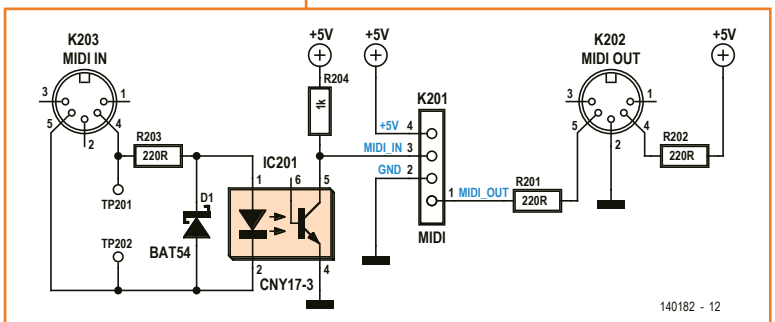
De hardware

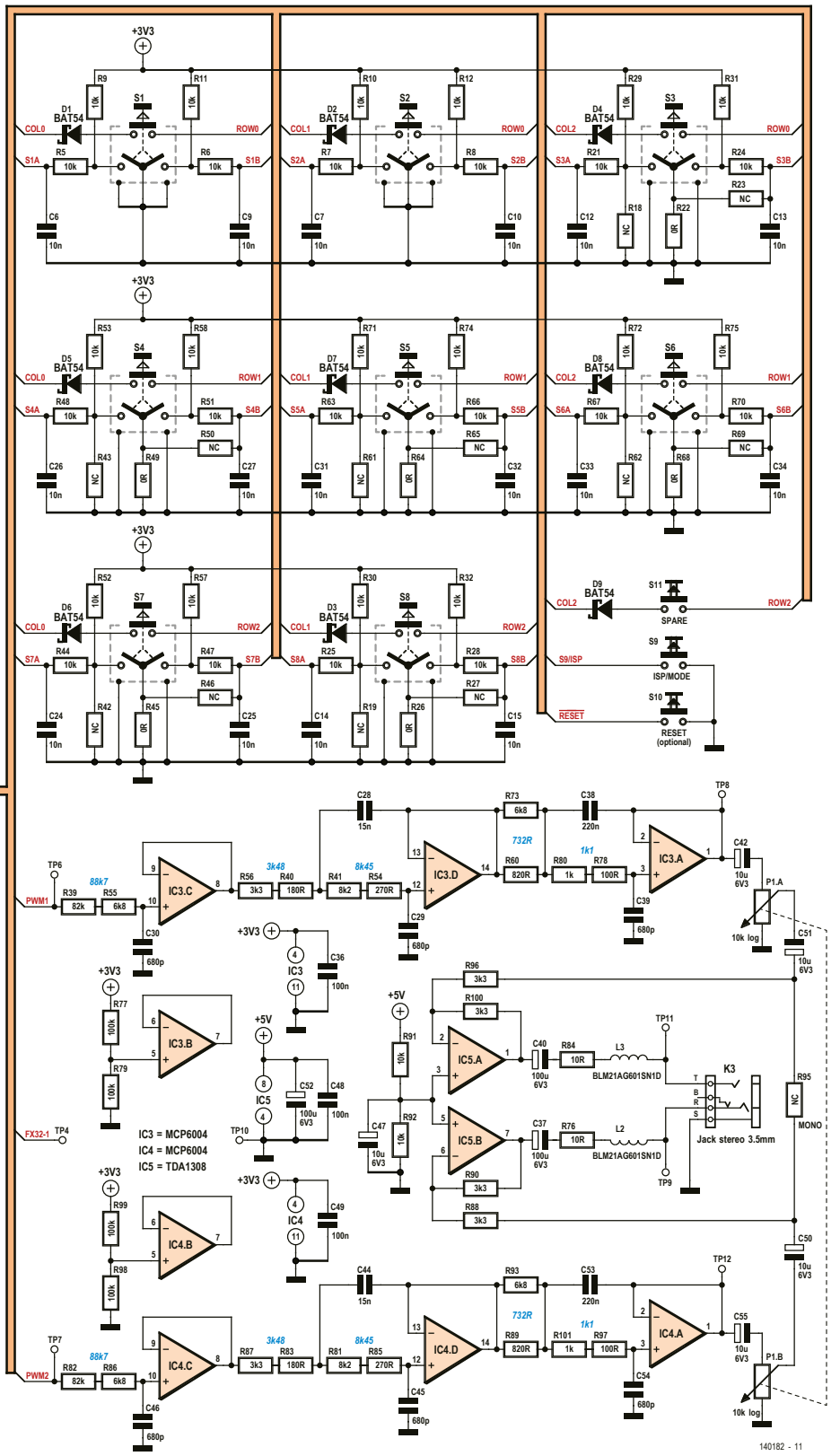
Toen ik eenmaal een weliswaar niet volledig functioneel, maar toch werkend prototype op mijn J²B-board (figuur 3) draaiende had, was het tijd voor het hardware-ontwerp van de synthesizer. Het plan was om het J²B-board in te zetten als brein. Bij mijn experimenten was echter al gebleken dat de bediening van de encoders op de print niet zo ergonomisch was: ze zaten te dicht op elkaar.
Nu moest ik sowieso nog een extra print ontwerpen voor het anti-aliasing-filter, de MIDI-interface, de hoofdtelefoonversterker en een EEPROM. Daarom besloot ik om een groot moederbord te ontwerpen waar de encoders ruimer op zouden passen en waar je dan ook het J²B-board op kon prikken. Dat moederbord zou dan meteen groot genoeg zijn voor *through-hole*-componenten uit onze *Elektor Labs Preferred Parts* (ELPP) bibliotheek.
Het print-ontwerp was snel klaar. Het vijfde-orde Chebyshev anti-aliasing-filter is voor me uitgerekend door FilterLab (een gratis utility van Microchip), met een kantelfrequentie van 15 kHz, want daarboven hoor ik toch niks. Dat kostte ongeveer 30 seconden. De rest van de print kostte wat meer tijd, maar een dag of tien later (voornamelijk de levertijd van de print) kon ik mijn nieuwe prototype gaan opbouwen.
Helaas drong toen pas tot me door dat mijn aanpak van de hardware bepaald rampzalig was. Er waren veel te veel verbindingen tussen het J²B-board en het moederbord nodig, en tot overmaat van ramp kon ik ook vrijwel niet bij de connectors. Na een paar onzinnige pogingen om mijn eer te redden gaf ik het op.
Dus wat had ik nu? Een mechanisch onbruikbaar ontwerp met een geheugenprobleempje. Om het goed te krijgen moest ik het J²B-board aan de kant zetten en de print opnieuw ontwerpen. Dat betekende dat er geen reden meer was om de LPC1343 te handhaven. Intussen is er namelijk iets nieuwers verschenen, de LPC1347, met tweemaal zoveel geheugen en een 4 KB EEPROM aan boord, met nog wel dezelfde uitstekende ingebouwde USB-bootloader. Dus waarom zou ik niet overstappen naar de nieuwere LPC1347?

Figuur 4.
Schema van de hoofdprint van de J²B-synthesizer.



Figuur 5.
Schema van de MIDI-interface. Vanwege de afmetingen van de DIN-connectors is deze ondergebracht op een aparte print. Het mechanische ontwerp wordt zo een stuk flexibeler.




+3V3
S1
S2
S3
S4
S5
S6
S7
S8
D1 BAT54
D2 BAT54
D4 BAT54
D5 BAT54
D7 BAT54
D8 BAT54
D6 BAT54
D3 BAT54
D9 BAT54
COL0
COL1
COL2
ROW0
ROW1
ROW2
S1A
S1B
S2A
S2B
S3A
S3B
S4A
S4B
S5A
S5B
S6A
S6B
S7A
S7B
S8A
S8B
S11
SPARE
S9
S9/ISP
ISP/MODE
S10
RESET
RESET (optional)
IC3.A
IC3.B
IC3.C
IC3.D
IC4.A
IC4.B
IC4.C
IC4.D
IC5.A
IC5.B
IC3 = MCP6004
IC4 = MCP6004
IC5 = TDA1308
PWM1
PWM2
FX32-1
Jack stereo 3.5mm
MONO
P1.A
P1.B
10k log
BLM21AG601SN1D
88k7
3k48
8k45
732R
1k1
140182 - 11

Figuur 6.
De J²B-synthesizer in een plexiglazen behuizing, gemaakt met een lasersnijder.

Mk. II

Mijn nieuwe ontwerp (V2 of Mk. II) is dus gebaseerd op een LPC1347. Behalve de connectors, de encoders en de LED's zijn alle componenten SMD's. Vanwege de symmetrie heb ik een tweekleuren-LED toegevoegd. Bij het porten van de code had ik ook nog ontdekt dat de Atmegatron een LED heeft die MIDI-activiteit weergeeft, (daar had ik overheen gelezen in de handleiding), dus die heb ik toegevoegd - dat is drie. De LPC1347 beschikt over meerdere kanalen per PWM-timer (de LPC1343 trouwens ook), daarom heb ik een tweede anti-aliasing-filter toegevoegd, zodat tweekanaals-klank mogelijk wordt. We gebruiken tenslotte een 32-bits Cortex-M3 op 72 MHz, dus waarom zouden we niet het onderste uit de kan proberen te halen?

Schema

In het schema van de synthesizer (**figuur 4**) zult u geen grote verrassingen zien. Het is me gelukt om alle beschikbare I/O-poorten te benutten. De acht draai-encoders bezetten er 16, twee per encoder. De weerstanden rond de zes encoders zijn gekozen met het oog op flexibiliteit. Met de juiste weerstandswaardes ontstaat nu zelfs de mogelijkheid om een encoder te vervangen door een potmeter, want één kant van de encoders

Onderdelenlijst

Hoofdprint

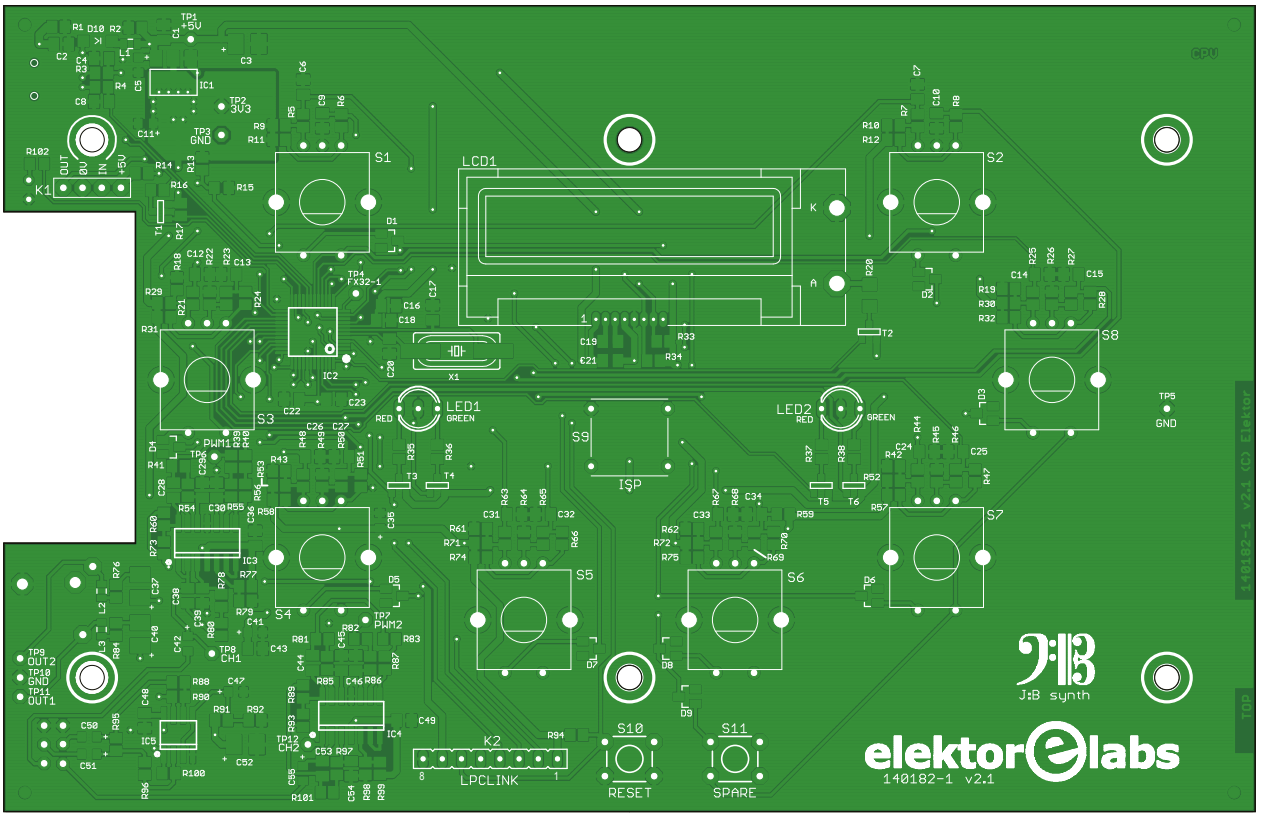
Weerstanden:
Alles SMD 0805, 5%/0,1 W, tenzij anders vermeld
R1,R22,R26,R45,R49,R64,R68 = 0 Ω
R2,R13,R59,R77,R79,R94,R98,R99 = 100 k
R3,R4 = 33 Ω
R5..R12,R21,R24,R25,R28..R32,R44,R47,R48,R51,R52,R53,R57,R58, R63,R66,R67,R70..R75,R91,R92 = 10 k
R14,R80,R101 = 1 k
R16 = 1k5
R17 = 1 M
R20 = 47 Ω, SMD 1206, 0,25 W
R33,R34 = 4k7
R35,R37,R54,R85,R102 = 270 Ω
R36,R38 = 220 Ω
R39,R82 = 82 k
R40,R83 = 180 Ω
R41,R81 = 8k2
R55,R73,R86,R93 = 6k8
R56,R87,R88,R90,R96,R100 = 3k3
R60,R89 = 820 Ω
R78,R97 = 100 Ω
R76,R84 = 10 Ω
P1 = 10 k potentiometer, stereo, logaritmisch
! R15,R18,R19,R23,R27,R42,R43,R46,R50,R61,R62,R65,R69,R95 = niet geplaatst

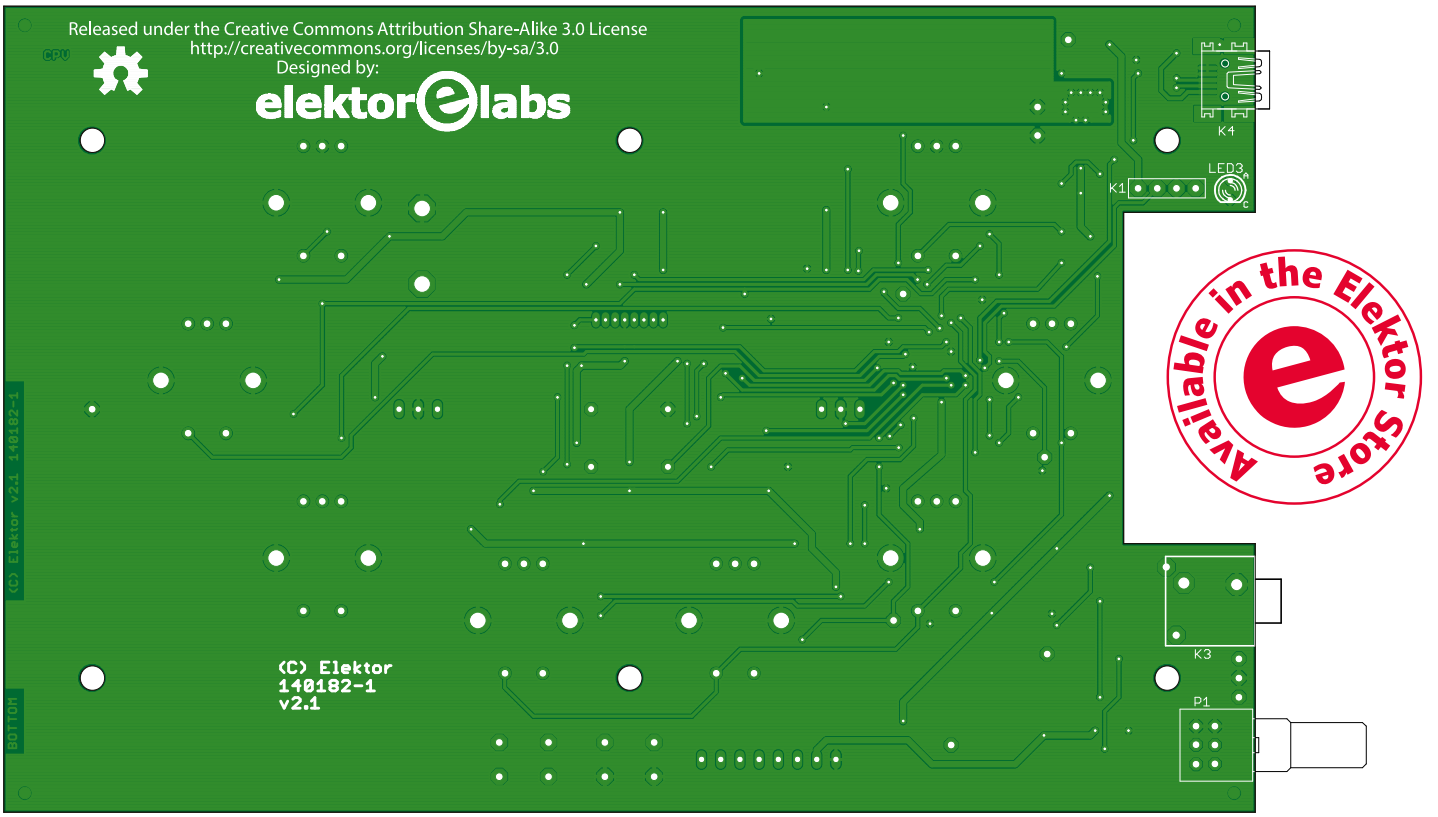
Condensatoren:
Alles SMD 0805
C1,C16,C22,C36,C48,C49 = 100 n
C3,C37,C40,C52 = 100 µ/ 6,3 V tantaal, maat B
C4,C8,C17,C20 = 18 p
C5,C11,C35,C41,C42,C47,C50,C51,C55 = 10 µ/6,3 V tantaal
C6,C7,C9,C10,C12..C15,C18,C23..C27,C31..C34,C43 = 10 n
C19,C21 = 1 µ
C28,C44 = 15 n
C29,C30,C39,C45,C46,C54 = 680 p
C38,C53 = 220 n
! C2 = niet geplaatst

Zelfinducties:
L1,L2,L3 = ferrietkraal, 0,21Ω/0,6 A, SMD 0805

Halfgeleiders:
D1..D9 = BAT54C (SOT-23)
D10 = MBR0540T1G
IC2 = LPC1347FBD48
IC1 = LD1117S33CTR
IC3,IC4 = MCP6004-I/SL
IC5 = TDA1308T/N2
LED1,LED2 = tweekleuren-LED rood/groen, CC, 5 mm
LED3 = LED rood, 3 mm
T1..T6 = BSS84P (SOT-23)

Diversen:
K1 = 4-pens pinheader, steek 2,54 mm
K2 = 8-pens pinheader, steek 2,54 mm
K3 = 3,5 mm klinkstekerbus, stereo
K4 = mini USB-B-connector, afgeschermd
S1..S8 = draai-encoder
S9 = druktoets, Multimec 3FTL6
LCD1 = LCD 2x16, I²C, bijv. Midas MCCOG21605B6W-SPTLYI
X1 = kristal 12 MHz
BOX1 = bijv. Hammond type 1597DGY
Hoofdprint nr. 140182-1


LCD1
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
LED1
LED2
ISP
LPCLINK
RESET
SPARE
J:B synth
elektor labs
140182-1 v2.1


Released under the Creative Commons Attribution Share-Alike 3.0 License
http://creativecommons.org/licenses/by-sa/3.0
Designed by:
elektor labs
(C) Elektor
140182-1
v2.1
K1
K3
K4
P1
LED3
Available in the Elektor Store

is verbonden met de ingang van de analoog/digitaal-converter (ADC) op de µC. Zie bijvoorbeeld S5: Voor een encoder plaatsen we R63, R64, R66, R71, R74, C31 en C32. Voor een potmeter plaatsen we R61, R65 en R74 (0 Ω) en misschien C32. De encoders zijn voorzien van een geïntegreerde druktoets. Die vormen met elkaar een toetsen-matrix van 3 x 3. De negende toets is reserve en doet momenteel geen dienst (maar zit wel op de print). Er zijn twee aparte toetsen voor reset en voor modus-keuze (rode of groene modus). Deze laatste wordt ook gebruikt voor het updaten van de firmware in de µC. De rood/groene tweekleuren-LED's worden aangestuurd door MOSFET's die een hoop stroom door de LED's kunnen sturen zonder dat de µC daar schade van ondervindt.

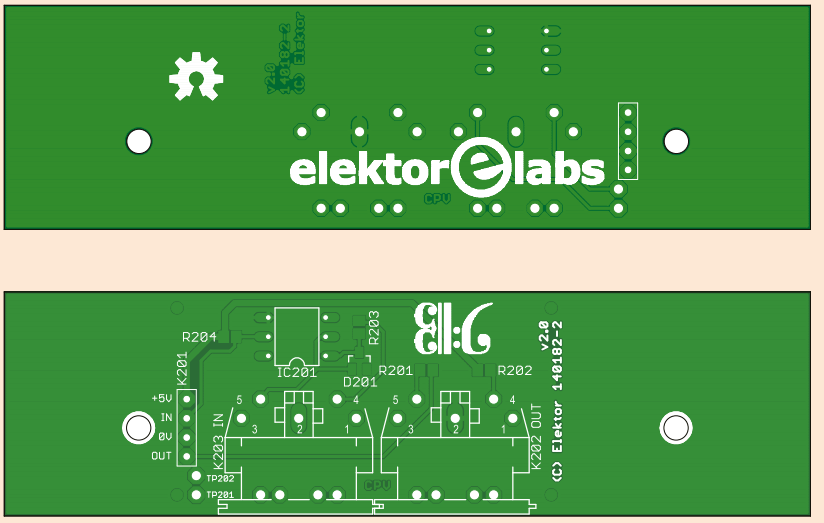
Het LCD is een I²C-type en ik vind het geweldig. Het is niet alleen kleiner met de perfecte hoogte voor de encoders, het is ook goedkoper dan normale LCD's en bezet minder pootjes op de µC.

De anti-aliasing-filters bestaan ieder uit drie opamps met een aantal weerstanden en condensatoren. De weerstanden zijn in tweeën gesplitst, zodat we met waardes uit de E12-reeks toch de vereiste niet-standaard waardes voor het filter kunnen samenstellen. Twee opamps in IC5 worden niet gebruikt. De uitgangen van de filters gaan via koppelcondensatoren naar de volumepotmeters om gekraak te voorkomen. De audiofielen onder u zullen nu wel het hoofd schudden, maar voor deze toepassing voldoet het prima. De stereo hoofdtelefoonversterker levert voldoende uitgangsvermogen voor de meeste toepassingen.

De MIDI-interface (**figuur 5**) bevindt zich op een aparte print omdat de DIN-connectors te hoog zijn voor de behuizing die ik gekozen had. Nu kunnen ze lager worden gemonteerd en past het perfect. Dit keer heb ik een net ontwerp gemaakt, zelfs met een boormal [2]. Ook heb ik voor het eerst met een lasersnijder gespeeld, waarvoor ik een tweede ontwerp moest maken (**figuur 6**). Deze tekeningen kunt u downloaden via [2].

Meer code overzetten

Met nieuwe hardware en een nieuwe microcontroller ontkwam ik niet aan een tweede ronde code-porten. De nieuwe en de oude µC behoren tot dezelfde familie, dus men zou denken dat het overzetten een kwestie van minuten was. Ik tenminste wel. Op het eerste gezicht hoef je alleen maar een paar signaalwegen aan te passen. In werkelijkheid was het wat ingewikkelder. NXP heeft de library voor de chip herschreven, omdat die qua architectuur meer op de LPC11Cxx-familie lijkt dan op de LPC134x-familie. Om kort te gaan: De LPC1347 is niet pen-compatibel en ook niet 100% code-compatibel met de LPC1343. Ik was nu al zo ver dat ik maar door de zure appel heen



Onderdelenlijst

MIDI-print

Weerstanden:
Alles SMD 0805, 5%/0,1 W
R201,R202,R203 = 220 Ω
R204 = 1 k

Halfgeleiders:
D201 = BAT54C (SOT-23)
IC201 = CNY17-3 (DIP-6)

Diversen:
K201 = 4-weg pinheader-connector, steek 2,54 mm
K202,K203 = 5-polige DIN-connector voor printmontage, 180°
MIDI-print nr. 140182-2

Available in the Elektor Store

gebeten heb en alle compatibiliteitsproblemen heb opgelost.

Doe het zelf

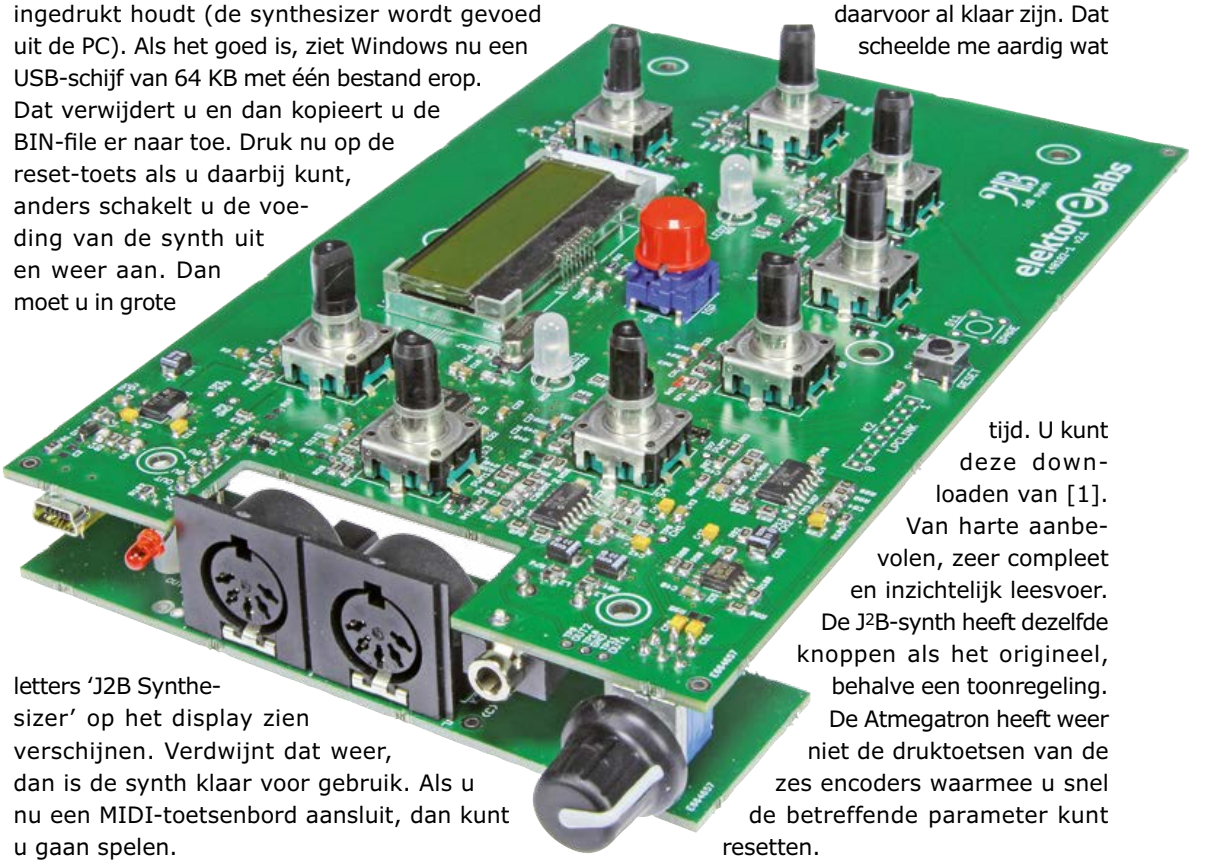
Het LPCXpresso/Eclipse software-project voor de synthesizer bestaat uit drie sub-projecten. Een voor de chip-library, een voor de board-library en een voor de synthesizer-applicatie zelf. Dat is iets complexer dan nodig, maar zo is het nu eenmaal. De pakketten zijn te importeren als zip-bestand (eerst uitpakken hoeft niet). Is alles met succes gecompileerd, dan vindt u een BIN-bestand in de map Release van het synthesizer-project. Om nu de µC te programmeren sluit u de synthesizer aan op een vrije USB-poort van een Windows PC terwijl u de rood/groene modus-knop ingedrukt houdt (de synthesizer wordt gevoed uit de PC). Als het goed is, ziet Windows nu een USB-schijf van 64 KB met één bestand erop. Dat verwijdert u en dan kopieert u de BIN-file er naar toe. Druk nu op de reset-toets als u daarbij kunt, anders schakelt u de voeding van de synth uit en weer aan. Dan moet u in grote letters 'J2B Synthesizer' op het display zien verschijnen. Verdwijnt dat weer, dan is de synth klaar voor gebruik. Als u nu een MIDI-toetsenbord aansluit, dan kunt u gaan spelen.

Alles is open

Dit project is 100% open-source en open-hardware. Alle ontwerp-bestanden van hardware, software en mechanica zijn gratis te downloaden via [2]. U bent van harte welkom om dit alles te bekijken. Schroom vooral niet om verbeteringen aan te brengen, want daar is op meerdere fronten ruimte voor. Ik ben heel benieuwd hoe dit project ontvangen wordt. Bouwt u een eigen exemplaar, dan zou ik daar graag een foto van ontvangen of een bijdrage op de webpagina voor dit project [2].

Handleiding

Het voordeel van het porten van de Atmegatron is dat de handleiding en de Librarian-utility daarvoor al klaar zijn. Dat scheelde me aardig wat tijd. U kunt deze downloaden van [1]. Van harte aanbevolen, zeer compleet en inzichtelijk leesvoer. De J²B-synth heeft dezelfde knoppen als het origineel, behalve een toonregeling. De Atmegatron heeft weer niet de druktoetsen van de zes encoders waarmee u snel de betreffende parameter kunt resetten.



(140182)

Weblinks

[1] Atmegatron: http://soulsbysynths.com/

[2] Projectpagina: www.elektor-magazine.com/140182

[3] J²B: Universele HMI-module met ARM Cortex-M3. Elektor september 2011, www.elektor-magazine.nl/110274; www.elektor-labs.com/node/3832