[Analog • Digital • Embedded • Mikrocontroller • Audio • Messtechnik]

G3078 E November 2011 (D) € 7,40 • CHF 14,30 • (A, B, L) € 8,20

elektor

Alles unter Kontrolle?

Low-cost-Strahlungs-messer

Alpha- / Beta- / Gamma-Zähler

Super-Arduino

Erste Schritte mit dem chipKIT™ Max32

DesignSpark chipKIT™ Challenge

Mini-DMX-Mischpult

Farbsteuerung für LED-Spot

Ultraschall-Konverter

Empfindlicher Fledermaus-Detektor

4 198630 307401 11

Von den Machern von Elektor!

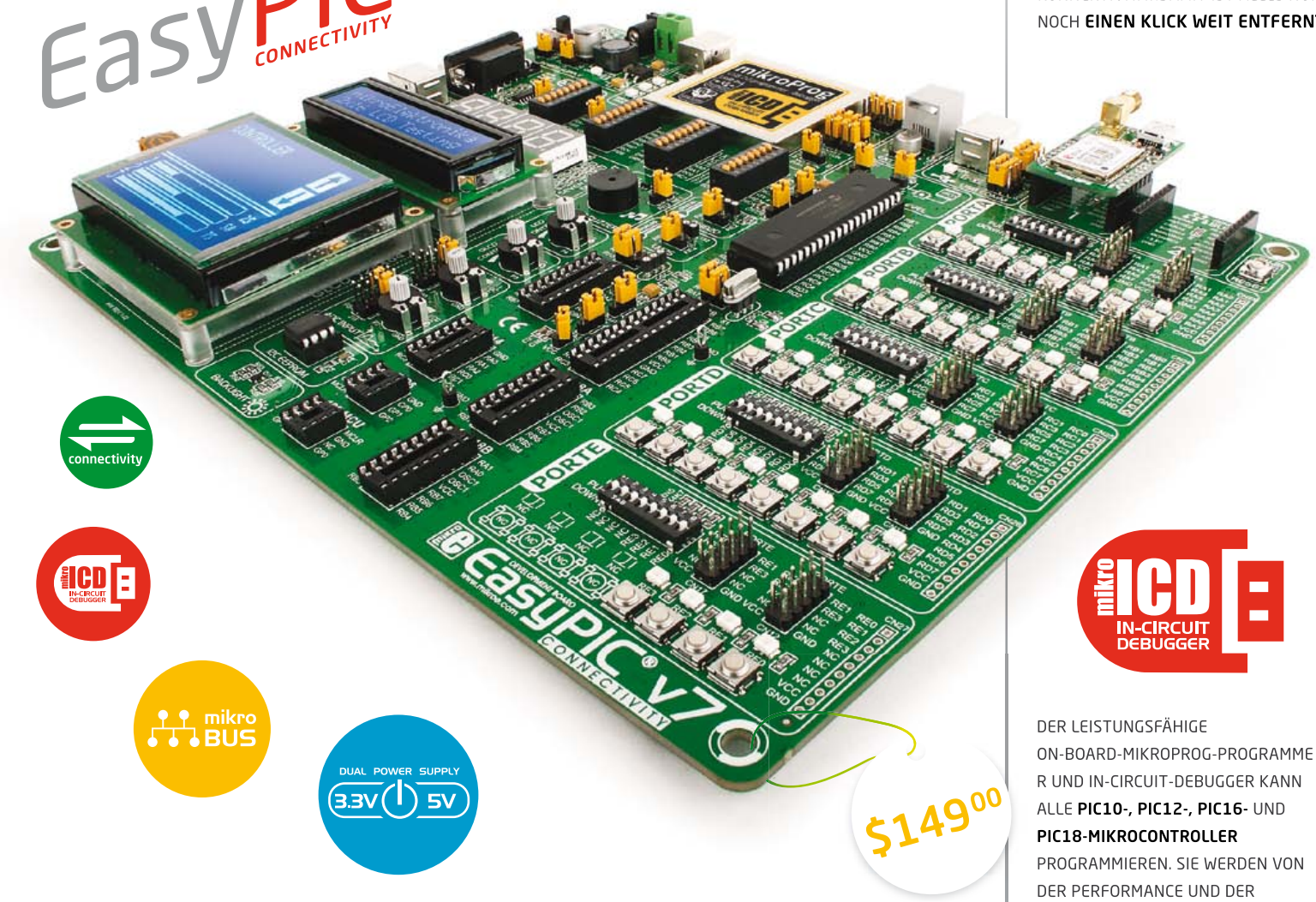
Oder frei Haus unter www.elektor.de bestellen!

IT'S HERE!

mikro BUS

MAN BRAUCHT NUR EIN **CLICK-BOARD™** ANSTECKEN UND SCHÖN LÄUFT ES. DAS SPEZIELLE **MIKROBUS™ -KOMMUNIKATIONS-INTERFACE** VEREINFACHT DIE ENTWICKLUNG UND ERMÖGLICHT SIMPLE UND DOCH HOCHEFFEKTIVE KONNEKTIVITÄT.DAMIT IST ALLES NUR NOCH **EINEN KLICK WEIT ENTFERNT!**

EasyPIC v7 CONNECTIVITY



connectivity

mikro ICD IN-CIRCUIT DEBUGGER

mikro BUS

DUAL POWER SUPPLY 3.3V 5V

149^{00}

mikro ICD IN-CIRCUIT DEBUGGER

DER LEISTUNGSFÄHIGE ON-BOARD-MIKROPROG-PROGRAMME R UND IN-CIRCUIT-DEBUGGER KANN ALLE **PIC10-, PIC12-, PIC16-** UND **PIC18-MIKROCONTROLLER** PROGRAMMIEREN. SIE WERDEN VON DER PERFORMANCE UND DER EINFACHEN FUNKTION VERBLÜFFT SEIN.OB ANFÄNGER ODER PROFI – EINMAL DAMIT GEARBEITET WILL MAN ES NICHT MEHR MISSEN.

DUAL POWER SUPPLY 3.3V 5V

EASYPIC V7 IST DERZEIT WOHL DAS EINZIGE-ENTWICKLUNGS-BOARD, DAS SOWOHL 3,3-V- ALS AUCH 5-V-MIKROCONTROLLER UNTERSTÜTZT. REVOLUTIONÄRE VERFAHREN FÜHREN ZUR UNTERSTÜTZUNG **VON 250 MIKROCONTROLLERN** DURCH EIN EINZIGES BOARD. ES IST, WIE WENN MEHRERE BOARDS IN EINEM STECKEN WÜRDEN!

Der weltweite Bestseller unter den PIC-Entwicklungs-Boardserscheint nun in dersiebten verbesserten Version und ist Spitzentechnik in Funktion und Qualität. Mitvier Steckverbindern für jeden Port von EasyPIC v7 ist es sehr **vielfältig anschließbar**. Die Ports sind logisch zu Gruppen für LEDs und Taster zusammengefasst. Der leistungsfähigeintegrierte **mikroProg**-In-Circuit-Debugger und-Programmer unterstützt über 250Chips mit **3,3 V und 5 V**. Ausgestattet mit zwei Display-Typen,seriellem EEPROM, zweiTemperatursensoren, Piezo-Buzzer, USB,RS-232 und FTDI, Oszilloskop-GND-Pins sowie **mikroBus-Support** ist dieses Board die unersetzliche Basis einer effektiven PIC-Entwicklungs-Umgebung.

MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

GREIFEN SIE ZU!
www.mikroe.com

elektor

Rück- und Vorschau

Seit einigen Monaten läuft unser Audio-DSP-Kurs in Elektor. Viele Elektroniker und Audio-Fans sind daran interessiert, weil DSPs sehr viele und attraktive Anwendungsmöglichkeiten bieten. Andererseits müssen wir aber zugeben, dass der Kurs bisher viel Theorie und wenig Praxis vermittelt hat. Das von einigen Lesern schon ungeduldig erwartete DSP-Board haben wir zwar inzwischen veröffentlicht und im vorliegenden Heft mit einem USB-kompatiblen Programmieradapter ergänzt, es fehlen aber immer noch konkrete Anwendungen.
Leider zählen DSPs nicht gerade zu den Convenience-Produkten der Elektroniker-Küche. Kein ready to go, kein plug and play: Hier muss man noch richtig kochen lernen, und der Kochkurs für diese komplexen Bauteile muss zuerst einmal die nötigen Grundkenntnisse vermitteln. Wer DSPs anwenden möchte, muss in diesen sauren Apfel beißen (und sich durchbeißen). Jedenfalls noch bei der aktuellen Kurs-Folge in diesem Heft. Doch schon ab der nächsten Ausgabe wird die Mühe mit praktischen Anwendungsbeispielen belohnt, und Sie müssen nicht mehr (nur) lesen und studieren, sondern können endlich auch selbst experimentieren.
Was es sonst noch im nächsten Heft gibt, steht wie immer auf der „letzten“ Seite (für die Redaktion ist das die Seite 84).
Was dort noch nicht steht, ist ein neuer internationaler Design-Wettbewerb in Kooperation mit RS Components, der zusammen mit unserer Schwesterzeitschrift Circuit Cellar in den USA ausgeschrieben wird. Für die Teilnahme am Wettbewerb kann man sich dann um einen Digilent chipKIT Max32 bewerben, und die Aufgabe wird darin bestehen, mit diesem Board und der DesignSpark PCB-Design-Software ein Projekt zu entwickeln, das dabei hilft, Energie zu sparen. Der offizielle Wettbwerbsbeginn wird er 26. November sein, und bis dahin haben Sie die Ausschreibung bereits mit dem Dezember-Heft erhalten. Über erste Schritte mit dem chipKIT Max32 informiert Sie Clemens Valens von der französischen Elektor-Ausgabe schon jetzt – auf Seite 14.

Ernst Krempelsauer

INHALT

42. Jahrgang
November 2011
Nr. 491

18 Low-cost-Strahlungsmesser
Der Alpha-/Beta-/Gamma-Zähler

Für die Messung radioaktiver Strahlung braucht man nicht viel mehr als eine PIN-Fotodiode und den passenden Sensorverstärker. Hier wird ein optimierter Vorverstärker mit einem Mikrocontroller-Zähler vorgestellt. Der Controller übernimmt auch gleich die Zeitmessung und zeigt die Impulsrate in „counts per minute“ an. Das Gerät kann mit unterschiedlichen Sensoren für Gamma- und Alphastrahlung verwendet werden. Es eignet sich gut für Langzeitmessungen und für Untersuchungen an schwach strahlenden Proben.

24 Hier kommt der Bus
Schnell zur eigenen Steuerung

Wer eine eigene Bus-Anwendung ausprobieren wollte, musste sich bisher mit kleinen Änderungen unserer Demos zufrieden geben - oder alles neu programmieren. Das wird nun anders: Die eigene Bus-Steuerung und ein maßgeschneidertes User-Interface sind rasch erstellt und bei Bedarf im Nu erweitert. Und da das Konzept auf HTML und Javascript basiert, kann dieselbe Zentrale auf so unterschiedlichen Plattformen wie PC und Smartphone zum Einsatz kommen.

32 Gradwanderung
Ein Messgerät für Temperaturänderungen

Manchmal interessieren nicht so sehr die Absolutwerte der Temperatur, sondern kleinste Temperaturänderungen - etwa bei Elektronik-Bauteilen oder Rohrleitungen. Die hier vorgestellte Schaltung spürt diese Änderungen nicht nur auf, sondern macht sie auch sicht- und sogar hörbar. Ein ADC mit SPI-Interface sorgt für eine Auflösung von einem zehntausendstel Grad!

40 Elektronische Stimme
Wollen Sie ihrem nächsten Projekt das Sprechen beibringen?

Manchmal ist es schwierig, Meldungen auf einem Display abzulesen, zum Beispiel, wenn man sein Modell-Flugzeug, -Schiff oder -Auto steuern muss, das mit Telemetrie ausgestattet ist. In solchen Fällen kann es eine gute Idee sein, sich über wichtige Parameter wie Geschwindigkeit, Höhe oder Akkuspannung akustisch unterrichten zu lassen. Hier kommt eine Schaltung, die nicht nur Zahlen vorlesen kann!

elektor international media bv

Eine multimediale und interaktive Plattform für jeden Elektroniker - das bietet Elektor International Media. Ob Anfänger oder Fortgeschrittener, ob Student oder Professor, ob engagierter Profi oder leidenschaftlicher Hobbyist: Hier finden Sie wertvolle Informationen, Inspiration für die eigenen Entwicklungen, Unterstützung bei der Ausbildung und nicht zuletzt eine gute Portion Unterhaltung. Gedruckt und im Web. Analog und digital. In Theorie und Praxis.

IMPRESSUM

42. Jahrgang, Nr. 491 November 2010

Erscheinungsweise: 11 x jährlich (inkl. Doppelheft Juli/August)

Elektor möchte Menschen anregen, sich die Elektronik zu Eigen zu machen – durch die Präsentation von Projekten und das Aufzeigen von Entwicklungen in der Elektronik und technischen Informatik.

Elektor erscheint auch in Englisch, Französisch, Niederländisch, Spanisch und weiteren Sprachen.
ELEKTOR ist in über 50 Ländern erhältlich.

Verlag
Elektor-Verlag GmbH - Süsterfeldstraße 25, 52072 Aachen
Tel. 02 41/88 909-0 - Fax 02 41/88 909-77

Technische Fragen bitten wir per E-Mail an redaktion@elektor.de zu richten.

Internationale Chefredaktion **Wisse Hettinga**

Redaktion Elektor Deutschland
Ernst Krempelsauer (Chefredakteur, v.i.S.d.P.)
Jens Nickel
(E-Mail: redaktion@elektor.de)

Internationale Redaktion
Harry Baggen, Thijs Beckers, Jan Buiting, Eduardo Corral, Clemens Valens

Redaktionssekretariat **Hedwig Hennekens**

Labor/Technische Redaktion
Christian Vossen (Ltg.), Thijs Beckers, Ton Giesberts, Luc Lemmens, Raymond Vermeulen, Jan Visser

Grafische Gestaltung und Layout
Giel Dols, Mart Schroijen

Geschäftsführer/Herausgeber: **Don Akkermans**

Marketing/Vertrieb (Leitung): **Carlo van Nistelrooy**

Anzeigen (verantwortlich): **Irmgard Ditgens**
ID Medienservice
Tel. 05 11/61 65 95-0 - Fax 05 11/61 65 95-55
E-Mail: service@id-medienservice.de
Es gilt die Anzeigenpreisliste Nr. 40 ab 01.01.2010

Vertriebsgesellschaft: **IPS Pressevertrieb GmbH**
Postfach 12 11, 53334 Meckenheim
Tel. 0 22 25/88 01-0 - Fax 0 22 25/88 01-199

E-Mail: elektor@ips-pressevertrieb.de
Internet: www.ips-pressevertrieb.de

Vertrieb Österreich
Pressegroßvertrieb Salzburg/Anif - Niederalm 300
Tel. +43/62 46/37 21-0

Druck: Senefelder Misset, Doetinchem (NL)

ISSN 0932-5468

Nochmals zurück zur Erde

Labcenter-Beitrag in Elektor 06/2011, S. 45 (110396) und Mailbox-Beiträge in Elektor 09/2011, S.8

In den Mailbox-Beiträgen zur Schutzleiter-Testschaltung sind noch einige Fehler enthalten.

1. In Deutschland beträgt die Normspannung nicht 230/380 V, sondern 230/400 V ±10 % Die Spannung von 400 V ergibt sich aus dem Verkettungsfaktor (Wurzel 3 ≈ 1,73) also 230 V x 1,73 = 400 V. Die „380“ sind nur noch in Hochspannungsnetzen vorhanden (380 kV).
2. IT-Systeme (-Netze) werden fast ausschließlich in kleinen Industrienetzen, in Krankenhäusern (OP) und ähnlichen Anlagen verwendet. In deutschen Ortsnetzen ist das IT-System wegen den Schutzmaßnahmen (z.B. Berührungsspannung) nicht mehr erlaubt, weil die Stromkreise mit Steckdosen mit Fehlerstromschutzschaltern (RCD) geschützt werden müssen (DIN-VDE, TAB). Bei IT-Systemen funktionieren RCDs nicht. Hier muss eine Isolationsüberwachung mit Abschaltung errichtet werden. Zu dem TN-System und IT-System gehört noch das TT-System, das auch stark verbreitet ist. Hierzu eine kurze Erklärung der Netzformen:

TN-System (frz. Terre Neutre): Sternpunkt-Neutralleiter (N) ist am Trafo (Generator) geerdet (N+PE) und wird als PEN-Leiter zur Verbrauchsstelle geführt (TN-C), in der Verbrauchsstelle wird der PEN wieder aufgeteilt (TN-S) in N + PE. Der Fehler/Erd-Strom wird über den PE-Leiter / PEN-Leiter und über das Erdreich geführt. Erdungsschleifenwiderstand < 1 Ω.

TT-System (frz. Terre Terre): Sternpunkt-Neutralleiter (N) ist am Trafo (Generator) geerdet (PE+N), aber es wird nur der N-Leiter zur Verbrauchstelle geführt, an der eine eigene Erdungsanlage erstellt werden muss. Der Fehler/Erd-Strom wird über das Erdreich geführt. Erdungsschleifenwiderstand > 1 Ω.

IT-System (frz. Isolé Terre): Sternpunkt-Neutralleiter (N) ist am Trafo (Generator) isoliert und kann als 3-Leitersystem 3x400 V ohne N-Leiter oder als 4-Leitersystem 3x400 V + N-Leiter (L1/L2/L3 gegen N = 230 V) geführt werden. Erdungsschleifenwiderstand hochohmig. Die vorgeschriebene Schutzmaßnahme Fehlerstromschutzschaltung in Steckdosenstromkreisen ist in dieser Netzform nicht möglich, weil die Abschaltung erst bei zwei oder mehr Fehlern erfolgt.

Quelle: DIN-VDE-Vorschriften und TAB technische Anschluss Bedingungen der VNB Versorgungs-Netz-Betreiber ehem. EVU Elektro-Versorgungs-Unternehmen.

Hans-Joachim Ahlswede (Ahlswede Elektrotechnik)

Eigene Effekte für den FV-1

Digitales Multi-Effektgerät, Elektor 09/2010, S. 20 (090835)

Mich würde noch interessieren, wie man selbst Programme für den FV-1 schreibt beziehungsweise vor allem, wie man sie in den Chip hinein bekommt. Oder stehen die im EEPROM?

Jean-Claude Feltes

Die Effekte werden komplett im FV-1 Chip erzeugt. Der ATmega8 übernimmt ausschließlich Verwaltungsaufgaben wie zum Beispiel Bedienerschnittstelle, Auswahl von Programmen und Parametern und so weiter. Der FV-1 besitzt insgesamt 16 Effektalgorithmen (also Effekttypen wie Hall, Echo, Pitchshifter und so weiter). Acht dieser Effekte sind fest im FV-1 „eingebrannt“ und können nicht verändert werden. Die anderen acht liegen im EEPROM des FV-1, also in IC5. Dieses EEPROM kann natürlich verändert werden, um neue Effekte zu hinterlegen. So könnte beispielsweise der Rotor-Effekt durch einen Verzerrer ausgetauscht werden.

Um neue Effekte zu programmieren, bedient man sich des kostenlosen Assemblers für den FV-1, den man auf der Website des Herstellers findet. Auf dieser Website sind auch andere Effekte beziehungsweise Beispiele zu finden. Mit dem Assembler können diese Beispiele dann in ein Hex-File für das FV-1 EEPROM umgewandelt werden.

Eigene (vollkommen neue) Effekte zu entwerfen ist mit dem Assembler natürlich auch möglich. Zum Testen der FV-1 Programme empfehle ich dann das Starter-Kit „SPN1001-DEVB“. Damit können die eigenen Effekte direkt per USB übertragen und anschließend getestet werden. Dies ist nicht so umständlich wie das ständige Ziehen und Neuprogrammieren.

Hubert Bollig

Updates und Ergänzungen

Jedes Mikrojoule zählt - Low Power Design bei AVR-Schaltungen

Elektor 03/2010, S. 38 (090157)

Es gibt inzwischen weitere Application Notes auf www.atmel.com zu diesem Thema. Der Autor (Andreas Riedenauer, INELTEK Mitte GmbH) hat in einem Nachtrag die interessantesten neuen App-Notes ausgewählt und kurz vorgestellt. Diese Übersicht ist auf der Elektor-Webseite zum Artikel (www.elektor.de/090157) unter „Infos & Updates” zu finden.

MailBox

In dieser Rubrik veröffentlichen wir Kritik, Meinungen, Anregungen, Wünsche oder Fragen unserer Leser. Die Redaktion trifft die Auswahl und behält sich Kürzungen vor. Bitte geben Sie immer an, auf welchen Artikel und welche Ausgabe (Monat/Jahr) sich Ihr Schreiben oder Ihre Mail bezieht.

Sie erreichen uns per E-Mail redaktion@elektor.de, per Fax (02 41/88 909-77) oder unter der Anschrift:
Redaktion Elektor
Süsterfeldstr. 25 – 52072 Aachen

Von Phil Knurhahn

Neues Speicherprinzip für höhere Datendichte

Forscher der Max-Planck-Gesellschaft in Halle wollen mit winzigen Eiseninseln die Kapazität heutiger Datenspeicher um das 400-fache erhöhen. Jede dieser Nanoinseln (Bild: MPI für Mikrostrukturphysik) besteht aus etwa 300 Eisenatomen, die in zwei Atomlagen auf einer Kupferunterlage aufgebracht sind. Für die Speicherung von Information wird als „Schreibstift" ein Rastertunnelmikroskop verwendet, dessen Spitze auf einem einzelnen Atom endet. Dadurch entsteht ein gewaltiges elektrisches Feld von 1 Milliarde V/m, mit dem man die Eiseninsel vom ferromagnetischen in den antiferromagnetischen Zustand umschaltet und umgekehrt. Die Umschaltung durch ein elektrisches Feld ermöglicht erst diese hohe

Fokussierung – magnetische Felder würden im Nanomaßstab bereits die benachbarten Eiseninseln mit beeinflussen, was eine eindeutige Umschaltung unmöglich macht. Für das Auslesen der Information nutzt man die Strom-Spannungs-Kennlinie zwischen Rastertunnelmikroskop und Eiseninsel.

www.mpi-halle.de

Systemtechnik für Elektroautos

Den Elektroautos gehört die Zukunft. Forscher von 33 Fraunhofer-Instituten hatten sich daher im Projekt „Fraunhofer Systemforschung Elektromobilität" zusammengeschlossen. Am 30. Juli 2011 wurde das zweijährige Projekt abgeschlossen, am 2. September hat man auf der ATP-Teststrecke in Papenburg zwei Fahrzeuge für Probefahrten vorgestellt. Basis dieser beiden Elektroautos ist der Artega GT, ein zweisitziger Sportwagen. In der ersten Version wurden bereits käufliche Komponenten eingebaut und das

Zusammenspiel dieser Komponenten optimiert. In der zweiten Version stecken neu entwickelte Komponenten. Ein Beispiel dafür sind Radnabenmotoren: Sie wurden von Anfang an für den europäischen Markt ausgelegt. So hat man den Motorendurchmesser so gewählt, dass er in einer 15-Zoll-Felge Platz findet (Bild: Ingo Daute / Fraunhofer IFAM). Der gesamte Antriebsstrang – also Motor samt Mitteltunnel, Kardanwelle und Getriebe – wurde aus dem Auto heraus in die Radnaben gelegt, was völlig neue Fahrzeugkonzepte möglich macht. Bei einem Auto, das außen etwa die Größe eines VW Golfs hat, wäre der Innenraum dann etwa so groß wie bei einem Fahrzeug der Oberklasse. Weiterer Vorteil: Die Leistung wird an jedem Rad bereitgestellt: Das bietet höhere Sicherheit, da die Räder separat gebremst und beschleunigt werden können.

www.fraunhofer.de/presse/presseinformationen/Elektromobilitaet_Papenburg.jsp

Skyscraper aus Silizium

Mit dreidimensionalen Siliziumchips lassen sich ganze Systeme in nur einem Gehäuse verwirklichen. Zwei Wege gibt es: Zum einen Chips, die als einzelne, bei der Produktion übereinander gestapelte Silizium-Plättchen („Die") realisiert werden. Die einzelnen Ebenen werden über integrierte „Vias" senkrecht miteinander verbunden. De-facto sind dies noch zweidimensionale Chips, die aber über mehrere Funktionsebenen verfügen. Die andere Lösung zielt auf das nachträgliche Übereinander-Stapeln separater Chips ab, die dann jeweils spezielle Funktionen übernehmen. Auch solche Chips sind bereits am Markt, wobei die Zahl der Chipebenen sich meist auf den einstelligen Bereich erstreckt. IBM und der amerikanische Chemieriese 3M planen nun, die Zahl der zu stapelnden Chips drastisch zu erhöhen – auf zunächst einmal bis zu 100 Dies. Technisch könnte man damit die Leistungsfähigkeit der 3D-Chips um das Tausendfache erhöhen. Mobile Geräte ließen sich weiter verkleinern und mit mehr Möglichkeiten ausstatten. Das Problem: Den benötigten Kleber, der die Chipebenen bis auf ein tausendstel Millimeter genau gegeneinander positionieren muss, gibt es nicht. Die beiden Großunternehmen wollen ihn nun gemeinsam entwickeln. Dieser Kleber muss nämlich aus dem dicht gepackten Chipstapel Hitze abführen, damit wärmeempfindliche Bauelemente oder logische Schaltungen nicht gestört werden. Endziel dieser 3D-Technik wäre es, ganze Siliziumwafer zu Hunderten übereinander zu stapeln und so die Kosten der Massenproduktion von Einzelchips zu senken.

www.youtube.com/watch?v=rbj5vrXulDo

Plaudernde Roboter

„Chatbots" nennt das „Creative Machines Lab" der Cornell University zwei Bildschirm-Roboter, die sich miteinander unterhalten können (Bild: Jason Yosinski, Cornell Creative Machines Lab). 100.000 $ Preisgeld und eine Goldmedaille winken dem, der ein Computerprogramm entwickelt, das

Dialoge zwischen den Robotern ermöglicht. Wichtigste Grundbedingung: Man darf keinen Unterschied mehr zu einer menschlichen Unterhaltung feststellen. Der Preis ist schon seit 20 Jahren ausgeschrieben – und noch immer nicht abgerufen worden. Im Oktober dieses Jahres wird erneut der Wettbewerb um den „Loebner-Preis in Artificial Intelligence" ausgeschrieben: Wenn es den Entwicklern gelingt, bei zwei oder mehr Juroren den Eindruck entstehen zu lassen, dass sich hier zwei normale Menschen unterhalten, sind erst einmal 25.000 $ abzukassieren. Wenn dann auch noch der audiovisuelle Eindruck vermittelt werden kann, dass hier zwei menschliche Lebewesen auf den Monitoren miteinander plaudern, sind auch die 100.000 $ fällig.

www.youtube.com/watch?v=WnzlbyTZsQY

The industry's brightest minds come here to shine.

Every year one place brings together the innovators on technology's cutting edge.
And for those four days, Vegas glows brighter.

2012 International CES®
THE GLOBAL STAGE FOR INNOVATION

PRODUCED BY CEA®

Tuesday, January 10–Friday, January 13, 2012
Las Vegas, Nevada | CESweb.org

REGISTER NOW

Nachrichten aus Forschung und Technik, interessante Produkt-Neuheiten und vieles mehr findet man aktuell unter www.elektor.de!

True-RMS-Digital-Multimeter mit USB-Schnittstelle

Das neueste Mitglied der bei Conrad Electronic erhältlichen VC-900-Familie von Voltcraft, das True-RMS-Digital-Multimeter VC930, bietet diverse hilfreiche Zusatzfunktionen. Dazu gehören Tiefpassfilter zur präzisen Spannungs- und Frequenzmessung an Antriebssteuerungen und anderen elektrischen Geräten, bei denen die Grundfrequenz von Strom und Spannung von Oberwellen überlagert ist.

Das Multimeter kann Aufzeichnungen mehrerer Messungen im mobilen Betrieb auch unbeaufsichtigt und automatisch durchführen. Dafür besitzt das Gerät einen internen Speicher von bis zu 20.000 Messdaten. Über die optische USB-Schnittstelle lassen sich die Daten einfach und potentialfrei zum PC transferieren. Das große Display mit einem Anzeigeumfang von bis zu 40.000 Counts und automatischer Hintergrundbeleuchtung erlaubt auch in dunklen Umgebungen eine zeitgleiche Darstellung mehrerer Messungen. Das Digitalmultimeter VC930 bietet außerdem die Möglichkeit der Echteffektivmessung von Wechselspannung und Strom zur präzisen Darstellung von komplexen Signalen oder nichtlinearen Lasten. Im Relativwertmodus wird das Messergebnis besonders bei niederohmigen Messungen oder auch Kapazitätsmessungen automatisch um den Leitungswiderstand reduziert. Das Messgerät (Best.-Nr.: 124703) ist zum Preis von 320,11 Euro erhältlich (inkl. Mehrwertsteuer, zzgl. Versand).

www.conrad.de

Einfache Umrüstung von Geräten: USB-zu-Digital-Module im DB9-Format

Der USB-Spezialist FTDI bietet nun auch USB-zu-Digital-Schnittstellenmodule im Format eines DB9-Steckverbinders. Die neuen Module ermöglichen das Nachrüsten einer USB-Anbindung in Geräten, die bisher mit einem solchen 9-poligen Steckverbinder ausgestattet waren, um serielle Signale zu übertragen.

Die Module sind in den Varianten DB9-USB-D3-F (3,3 V, Buchse), DB9-USB-D3-M (3,3 V, Stecker), DB9-USB-D5-F (5 V, Buchse) und DB9-USB-D5-M (5 V, Stecker) erhältlich. Alle Module basieren auf dem bekannten USB-zu-Serial-UART-Bridge-Chip FT232R, der Datenraten bis zu 12 Mbit/s (USB Full Speed) unterstützt. Sie ermöglichen ein schnelles und einfaches Nachrüsten der Hardware auf den heute meistverwendeten seriellen Schnittstellenstandard – und das auf komfortable und kosteneffiziente Weise. Treiber für die Module stehen zum Download über die Website des Herstellers zur Verfügung.

Jedes Modul wird im kompakten 30,8 mm x 19,8 mm x 11,5 mm Gehäuse mit Standard-USB-Mini-B-Stecker ausgeliefert. Der Betriebstemperaturbereich erstreckt sich von -40 bis 85 °C, was auch einen Einsatz der Module in rauen Industrieumgebungen gestattet. Der Preis wird mit 15,50 US-$ (für 1-9 Stück) angegeben.

www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_DB9-USB.pdf

iPhone-/iPad-App von Reichelt

Über eine kostenlose iPhone- und iPad-App kann man nun von unterwegs auf den Shop von Reichelt zugreifen, auch eine Online-Bestellung ist möglich. Die leicht zu bedienende Applikation bietet schnelle und übersichtliche Suchergebnisse, was besonders dem Handwerker und Servicetechniker unterwegs entgegenkommt.

Dabei wird das gesamte, 35.000 Artikel umfassende Sortiment des norddeutschen Distributors abgebildet. Zu den Produkten sind detaillierte Ansichten mit technischen Daten und Zusatzinformationen abrufbar. Bestellungen werden über die App sicher und unkompliziert abgewickelt. Eine Version für Android-Smartphones und -Tablets ist in Vorbereitung.

www.reichelt.de

DESIGNSPARK

Turn a hot idea into a cool solution.

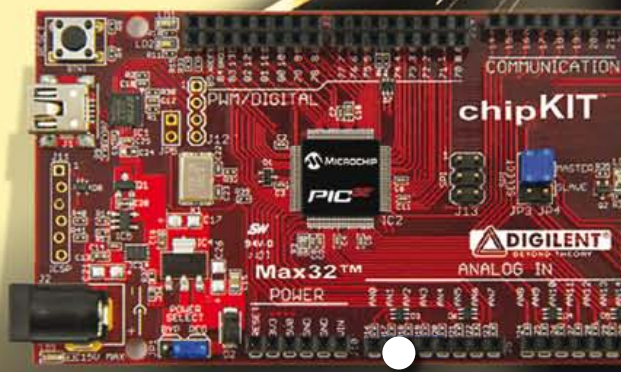
DesignSpark chipKIT™ Challenge

Contest launches **November 28, 2011!**

Challenge your talent against other engineers worldwide to produce an energy-efficient design solution using the free DesignSpark PCB software and Microchip's chipKIT™ development board.

Achieve the most energy-efficient design and **you could win a share of $10,000!**

Plus, keep the DesignSpark community regularly informed through posts on the DesignSpark Project Pages and your updates will make you eligible for Community Choice Awards and random prize drawings!



FREE chipKIT™ Max32™ development kit for qualified engineers.

Visit **www.chipkitchallenge.com**

for complete rules and to see if you qualify for a **FREE** ChipKIT™ Max32™ development kit.*

*Subject to availability.

IN ASSOCIATION WITH:

elektor · MICROCHIP · RS · CIRCUIT CELLAR · ALLIED ELECTRONICS · DIGILENT BEYOND THEORY

DesignSpark chipKIT™ Challenge

Super-Arduino

Erste Schritte mit dem chipKIT™ Max32

Von Clemens Valens (Elektor Frankreich)

Mikrocontroller-Interessierte werden vermutlich schon etwas von Arduino gehört oder sogar schon damit gearbeitet haben. Im letzteren Fall wird man auch schon die Grenzen dieser netten 8-bit-Plattform gestreift haben und sich mehr Rechenleistung, mehr Speicher oder mehr I/O-Pins wünschen. Es gibt zwar „bessere" Boards in Hülle und Fülle, allerdings muss man sich da wieder in andere Tools einarbeiten. Bisher. Nun aber hat Digilent eine Lösung mit mehr Leistung vorgestellt, für die man keine neuen Tools benötigt. Mit dem chipKIT™ Max32 bekommt man 32-bit-Technik und bis zu 80 I/O-Pins plus Kompatibilität zur Arduino-Welt.

Sicher gab es auch zuvor schon Versuche mit 32-bit-Arduino-Boards. Soweit mir bekannt ist, bezog sich deren Kompatibilität aber lediglich auf den Formfaktor und eben nicht auf die Tools. Für einige dieser Boards gibt es Libraries, die eine dem Arduino ähnliche Funktionalität und Syntax bieten, aber trotzdem einen anderen Compiler und ganz spezielle Techniken zum Laden von Firmware benötigen. Digilent hat die Arduino-Kompatibilität nun durch Integration der für PIC32-Controller nötigen Compiler, Linker und Programmer in die Arduino-IDE (**I**ntegrated **D**evelopment **E**nvironment) ein Stück weiter getrieben. Von der Arduino-IDE aus gesehen ist chipKIT lediglich ein Target, das neben den klassischen 8-bit-Arduino-Boards aufgelistet wird. Digilent ging sogar soweit, die zugehörige Webseite genau wie die von Arduino unter der Kennung „.cc" [1] zu registrieren. Ganz im Geiste von Arduino ist die chipKIT-Hardware (Schaltpläne und Layout-Dateien im Eagle-Format) und Software komplett Open Source und steht somit kostenlos im Internet zur Verfügung. Im Gegensatz zu Arduino handelt es sich bei der chipKIT-Platine um eine Vier-Layer-Ausführung, womit sich nur wenige trauen dürften, sich ihre eigene Chipkit-Platine zu machen.
Es gibt zwei chipKIT-Varianten: Uno32 und Max32. Mechanisch ist Uno32 mit Arduino Uno kompatibel und Max32 mit Arduino Mega, einer längeren Version des Standard-Arduino-Boards. Digilent-Boards sind allerdings rechteckig, um die eigentümliche Arduino-Form zu vermeiden – sie fallen daher etwas größer aus, was für kaum jemanden ein Problem darstellen dürfte.
Die weitere Beschreibung beschränkt sich auf Max32.

Das Board

Das Max32-Board ist eine rote Multilayer-Platine in einer rot/weißen Schachtel ohne viel Zubehör. Weder USB-Kabel noch Doku-

mentation ist dabei – nur ein Weblink [2]. Das Board selbst misst 102 x 54 mm. Wie beim Arduino-Mega sind drei Seiten der Platine mit Steckverbindungen bestückt. Abweichend davon ist der Steckverbinder für die digitalen Pins 0...13 (in Arduino-Notation) eine zweireihige Ausführung. Die digitalen Pins 70...85 liegen auf den Kontakten in der zweiten Reihe. Neben USB- und Power-Buchse gibt es noch Platz für den ICSP-Programmer-Anschluss (Microchip-spezifisch) in einer speziellen Sparkfun-Variante mit versetzten Kontakten. Ein gesteckter Pinheader bekommt so sicheren Kontakt, ohne eingelötet zu sein.

Das Board kann via USB oder extern über die Netzteil-Buchse versorgt werden, die für Gleichspannungen bis 15 V ausgelegt ist. Jumper JP1 überbrückt den 5-V-Spannungsregler, was bei Spannungen über 5 V gefährlich werden kann. Wenn man ein fabrikfrisches Board einschaltet, dann blendet eine aufdringlich helle roten LED neben der Netzteil-Buchse. Sie zeigt an, dass die 3,3-V-Schiene Spannung führt. Gleichzeitig blinkt eine grüne LED mit etwa 3 Hz.

Ein Max32 ist kaum mehr als ein BoB (**B**reak**o**ut-**B**oard) für einen 100-Pin-PIC32-Controller in der Mitte der Platine, der lediglich um Stromversorgung und USB-Anschluss ergänzt wurde. Der 32-bit-Mikrocontroller PIC32 von Microchip ist mit einem ARM-Cortex-M3 zu vergleichen (siehe Kasten). Das Board ist mit dem PIC32MX795F512L, dem größten verfügbaren Exemplar bestückt. Im 100-poligen Gehäuse stecken 512 KB Flash-Speicher und 128 KB RAM. Die CPU läuft mit 80 MHz. Neben USB-OTG (**O**n-**T**he-**G**o) gibt es einen Ethernet-MAC und zwei CAN-Controller.

Nach diesem Hardware-Überblick geht es zur Software:

Die IDE

Wie schon erwähnt, wird mit einer modifizierten Arduino-IDE entwickelt, die man kostenlos von [2] downloaden kann. Die Installation dieser 128-MB-Datei ist einfach: Man braucht sie nur zu entpacken (ausgepackt etwa 480 MB) und das Resultat an den gewünschten Platz auf der Platte kopieren. Die IDE wird mit Hilfe des Programms „mpide.exe“ gestartet, die sich im Wurzelverzeichnis des IDE-Ordners befindet. Die IDE arbeitet nicht nur unter Windows, sondern auch unter Linux und OS X. Allerdings muss zuvor Java installiert sein.

Durch die Integration der Tools in die Arduino-IDE hat sich Digilent eine Menge Dokumentationsarbeit gespart. Tatsächlich ist alles Wissenswerte zur Installation und Inbetriebnahme auf der Arduino-Website [3] schon beschrieben. Das beantwortet auch die Frage nach der Syntax der Programmiersprache.

Als dieser Artikel geschrieben wurde, war die IDE-Version 0022 (genauer: „mpide-0022-chipkit-win-20110619“) aktuell und damit auf dem gleichen Stand wie die offizielle Arduino-IDE. Laut Digilent besteht der einzige Unterschied in der Erweiterung durch den PIC32-Compiler/Linker nebst zugehörigen Libraries. Die IDE kann also nach wie vor auch zur Entwicklung von 8-bit-Arduino-Anwendungen leingesetzt werden. Ich musste leider feststellen, dass ausgerechnet mein Arduino-Clone (Seeeduino v1.1) nicht erkannt wurde: „Invalid device signature“ war zu lesen. Mit der offiziellen Arduino-IDE 0022 gab es keine Probleme.

Wissenswertes zum PIC32

Beim Stichwort „32-bit-Mikrocontroller“ denken die meisten Elektroniker wohl an ARM und dann vielleicht noch an Hersteller, die ARM-Technik in Silizium gießen (Atmel, ST oder NXP). Kaum jemand denkt an Microchip. Auch wenn viele Handys auf ARM-Technik basieren, so stecken doch in etlichen Geräten (Kameras und Drucker) eher Prozessoren auf MIPS-Basis. Ich habe es zwar nicht selbst überprüft, aber gemeinhin wird angenommen, dass auf der Welt mehr 32-bit-MIPS- als ARM-Controller existieren. Wenn man schon einmal mit MIPS-CPUs zu tun hatte, dann hilft das sicherlich auch beim PIC32, denn das ist ein prima Controller für den Einstieg in die 32-bit-Welt.

Gegenwärtig gibt es sogar fünf Familien: 3xx, 4xx, 5xx, 6xx und 7xx. Die Varianten 3xx und 4xx sind für allgemeine Anwendungen gedacht. Die restlichen drei Familien eignen sich für spezielle Zwecke wie CAN oder Ethernet und bringen teilweise mehr RAM mit. Die Chips basieren alle auf dem 32-bit-MIPS-Kern MK4, der mit einer fünfstufigen Pipeline Taktfrequenzen von bis zu 80 MHz erreicht. Als maximale Leistung werden 1,56 DMIPS/MHz (2,1 Dhrystone) angegeben, also etwas mehr als die 1,25 DMIPS/MHz eines ARM Cortex-M3.

Neben bis zu 512 KB Flash und 12 KB Boot-Memory kommen die Reihen 3xx und 4xx auf bis zu 32 kB RAM, während die restlichen Reihen mit bis zu 128 KB RAM aufwarten können. Die übliche Peripherie wie serielle Schnittstellen, PWM, ADC etc. ist natürlich eingebaut und zusätzlich gibt es mehrere DMA-Kanäle.

Die Chips sind in zwei Gehäuseformen mit 64 Pins (Suffix H) oder 100 Pins (Suffix L) zu haben. Ein 121-XBGA-Gehäuse enthält die 100-Pin-Variante. Die PIC32-Familie ist pinkompatibel mit einigen Chips der PIC24- und dsPIC-Reihe, sodass sie gut in die bisherige Palette von Microchip und zur gängigen IDE (MPLAB) passen. Auf der PIC32-Webseite von Microchip (www.mypic32.com) können Infos und Code für PIC32-Projekte ausgetauscht werden. Datenblätter und andere Informationen findet man unter „www.microchip.com/pic32“.

Family	USB OTG	CAN	Ethernet	RAM
3xx	–	–	–	≤ 32 KB
4xx	1	–	–	≤ 32 KB
5xx	1	1	–	≤ 128 KB
6xx	1	–	1	≤ 128 KB
7xx	1	2 (1 bei 764)	1	≤ 128 KB

Bild 1. Die Max32-IDE mit zu löschendem AVR-spezifischen „#define“.

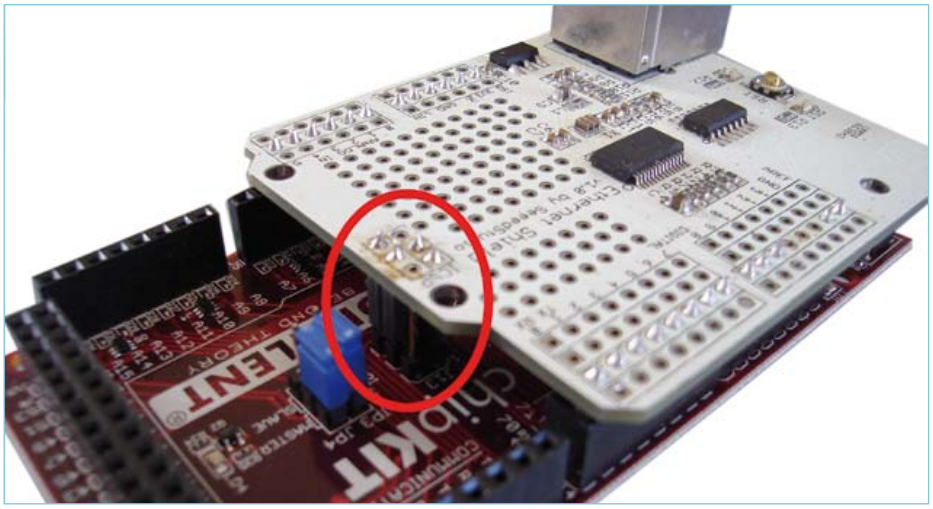


Bild 2. Blick auf die SPI-Verbindung von etherShield und Max32.

Nach Anschluss von Max32 an den PC und Installation der IDE kann man versuchen, eines der einfacheren mitgelieferten Beispiele zu kompilieren und auf das Board zu übertragen. Zuvor sollte das Max32-Board im Menü via Tools -> Board ausgewählt und die Parameter der seriellen Schnittstelle via Tools -> Serial Port korrekt eingestellt werden. Das Beispiel „BlinkWithoutDelay“ (File -> Examples -> Digital) sollte ohne Modifikation laufen und die grüne LED LD4 mit einer Frequenz von 0,5 Hz blinken lassen.
Wenn bis hierher alles geklappt hat, dann ist man bereit für die Entwicklung „richtiger“ Max32-Anwendungen. Hier gibt es allerdings noch die eine oder andere Hürde zu überspringen...

Shield-Portierung

Blinkende LEDs sind auf die Dauer nicht sehr befriedigend, weshalb ich mein Arduino-Ethernet-Shield am Max32 getestet habe (Shield = Arduino-Erweiterungskarte). Dieses Shield basiert auf Microchips ENC28J60-Ethernet-Controller mit SPI-Interface. Zwar hat ein PIC32 schon einen Ethernet-MAC integriert, aber ich hatte gerade kein Ethernet-PHY mit RJ45-Buchse zur Verfügung. Digilent empfiehlt solch ein Shield (das auch noch andere Dinge mitbringt), aber ich konnte nicht warten, bis mir ein solches geliefert werden würde. Außerdem konnte ich so gleich die Arduino-Kompatibilität testen. Ganz so kompatibel zeigte sich Max32 dann doch nicht...
Mein altes Ethernet-Shield – zur Unterscheidung vom offiziellen W5100-basierten Ethernet-Shield „etherShield“ genannt – wird lediglich von einer Bibliothek mit einigen Beispielen unterstützt. Der Code und etherShield funktionierten mit einem Seeeduino-Board problemlos. Von daher installierte ich zunächst diese Library in der Max32-IDE im Verzeichnis „libraries\“ und war gespannt, was sich nach Kompilierung tun würde. Und es tat sich - nichts! Der Grund hierfür lag nicht im Code selbst. Der Compiler hat schlicht nichts gefunden, was er hätte kompilieren können. Laut der Digilent-Webseite, auf der zumindest einige Aspekte der Portierung von Arduino-Code erläutert sind, werden Libraries gleich wie bei der Arduino-IDE behandelt. Nur eben nicht in meinem Fall. Als ich die Library dann in das Verzeichnis „Hardware\pic32\libraries\“ bewegte, wo sich die selben Dinge wie im Verzeichnis „libraries\“ befinden, fand der Compiler zwar den Code, produzierte aber viele Fehler und die Warnung, dass der Code AVR-spezifisch wäre. Arduino-Boards basieren auf Atmel-Controllern. Jetzt wusste ich zumindest, was los war.
Was bei der Portierung von Arduino-Libraries zuerst raus muss, das sind die Referenzen zum Programm-Speicher. Bei AVR-Controllern gibt es spezielle Compiler-Direktiven zum Ansprechen von Konstanten (Strings, Tabellen). Im Code für PIC32 haben diese nichts zu suchen. Um den Code aber Arduino-kompatibel zu halten ist es besser, diese Direktiven per „#define“ auszublenden. Hierfür gibt es das Makro „_BOARD_MEGA_“ (siehe **Bild 1**) der Max32-IDE. Eigentlich hätte man hierfür die Bezeichnung wie „_BOARD_MAX32_“ erwartet. Auch mit den AVR-spezifischen „#include“-Direktiven wird so verfahren. Manchmal – wie in meinem Fall – ist das noch nicht alles. Die Library hatte nämlich noch Bezüge auf AVR-Register, die ein PIC32 natürlich nicht kennt. Beim SPI-Treiber für den Ethernet-Chip ENC28J60 war dies der Fall, vermutlich weil er schon etwas älter ist und damals noch keine SPI-Library bei der Arduino-IDE dabei war (sie wurde erst in die Version 0019 vom September 2010 aufgenommen). Ich entschied mich, die etherShield-Library so zu ändern, dass sie die neue Arduino-SPI-Library nutzt und probierte dies zunächst auf einem Arduino-Board aus, bevor ich den Code auf das Max32-Board los ließ.
Dies führte zu neuen Problemen, da die SPI-Library nun Pin-Funktionen nutzte, die dem Max32-Compiler nicht gefallen. Es stellte sich heraus, dass sich meine in C geschriebene und so kompilierte Library nicht mit der in C++ erstellten SPI-Library vertrug. Dateien mit der Extension „.c“ werden als C-Code behandelt, und die mit der Endung „.cpp“ werden als C++-interpretiert. Ich musste also zunächst die komplette etherShield-Library nach C++ konvertieren und testen. Das war zwar nicht so schwierig, aber es zeigten sich Feinheiten wie an unerwarteten Stellen versteckte Compiler-Direktiven („#extern „C“ { ... }“). Nach diesen letzten Änderungen fand der Compiler der Max32-IDE im Code für das Max32-Board keine Fehler mehr. Aber funktioniert das auch?

Natürlich nicht! Das war zwar keine Überraschung, denn auf der Webseite von Digilent

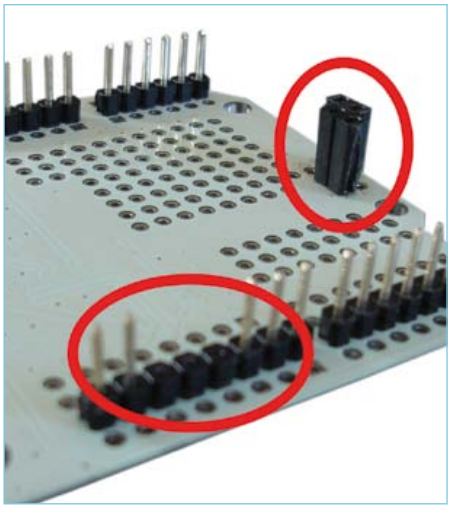


Bild 3. Meine Modifikationen beim etherShield.

Bild 4. Erfolgreiche Portierung: Ein kleiner Webserver auf einem Max32-Board.

bin ich schon auf einige gepostete Probleme mit der SPI-Anbindung gestoßen. Ich blieb dennoch optimistisch.

Die größten Probleme macht der Unterschied bei den I/O-Pins zwischen PIC32- und AVR-Controllern. Die digitalen Arduino-Pins 10...13 „können“ SPI und die Pins 10/11 sind zudem PWM-tauglich. Ein PIC32 kombiniert andere Pin-Funktionen als ein AVR-Controller. Digilent hat sich entschieden, der Kompatibilität der PWM-Funktionen den Vorzug zu geben, da sie für die Analog-Ausgaben mit Arduino-Boards häufiger verwendet werden. Deshalb ist hier nur ein Teil von SPI-Port 2 verfügbar. Allerdings fanden sie auch eine Methode, den SPI-Port 1 in Arduino-kompatibler Weise zugänglich zu machen, indem die Pins des ICSP-Ports (**Bild 2**) genutzt werden. Hier liegen die gleichen Signale wie an den Pins 10...13. Eigentlich ist mir diese Verbindung als Teil eines kompatiblen Shields nie in den Sinn gekommen, und da bin ich wohl nicht alleine. Doch Seeedstudio als Hersteller meines etherShields hat den ICSP-Pinheader exakt an der richtigen Stelle platziert. Das Ersetzen des Pinheaders durch eine entsprechende Buchse dauerte nur Minuten, und schon war eine SPI-Anschlussmöglichkeit an das Max32-Board gegeben. Um Konflikte mit den Pins 11...13 (MOSI, MISO & SCK) zu vermeiden, entfernte ich diese Pins an meinem Shield (siehe **Bild 3**). Jetzt aber sollte alles klappen, oder?

Schön wär’s gewesen. An diesem Punkt trat mein Oszilloskop auf den Plan, denn ich vermutete Inkompatibilitäten zwischen den vom PIC32 gelieferten ISP-Signalen und dem, was der ENC28J60 erwartet. Besonders die Taktrate irritierte mich. Und das Oszilloskop bestätigte meinen Verdacht: Da für einen Arduino 610 kHz schnell ist, serviert ihm ein PIC32 mit 20 MHz eine Überdosis. Das Datenblatt zum ENC28J60 meinte zwar, dass dies in Ordnung gehe, aber meine Experimente zeigten, dass 2,5 MHz deutlich realistischer sind. Zunächst reduzierte ich aber die Taktrate mit 625 kHz auf einen für Arduino passablen Wert - und war noch immer nicht am Ziel.

Doch ich stand kurz vor der Lösung. Heutzutage haben selbst die preiswertesten Oszilloskope eine Speicherfunktion – auch mein mit 240 € recht billiges 25-MHz-Modell Atten ADS1022C. Diese Funktion war sehr nützlich, denn ich entdeckte so Probleme mit der Phase/Polarität zwischen SPI-Takt und Datenleitungen. Es stellte sich heraus, dass das Max32-Board mit Mode 0 betrieben wurde - beim Shield aber Mode 1 erwartet wurde. War das jetzt die letzte Änderung? Ja - endlich! Siehe Beweisfoto in **Bild 4**.

Schlussbemerkung

Die Integration eines PIC32 in die Arduino-Umgebung durch Digilent ist schon eine tolle Sache, auch wenn sie nicht zu 100 % kompatibel ist. Digilent hat wohl das Machbare erreicht. Einfache Shields mit einfachen Libraries, die die Arduino-Design-Richtlinien einhalten, dürften wohl sehr einfach zu portieren sein. Man kann natürlich trotzdem auf einige Probleme auflaufen, wie mein Beispiel zeigt. Je mehr komplexe Shields Feinheiten von AVR-Controllern ausnutzen, desto schwieriger wird es werden. Man braucht dann schon einiges an PIC32-Wissen. Die Arbeit bleibt einfacher, wenn man sich soweit wie möglich an die Arduino-Libraries hält und damit Digilent die Portierungsarbeit überlässt.

Digilent sammelt alle bekanntermaßen funktionierenden Shields in einer Liste. Bevor man ein Projekt startet, sollte man hier nachschauen. Außerdem dürften Updates zur Max32-IDE mit der Zeit einige der hier aufgetretenen Probleme ausmerzen. Man sollte daher immer die neueste Version einsetzen.

Der einzige unklare Punkt ist, wo die eigenen Libraries abgelegt werden sollen. Nachdenken und Ausprobieren führten zum Schluss, dass wohl alle Libraries mit PIC32-Code (Low-Level-Treiber etc.) im Verzeichnis „Hardware\pic32\libraries\“ stecken sollten. Das gilt auch für Dateien, die von diesen Dateien benötigt werden. Alle anderen Dateien inklusive der Beispiele, die diese Dateien nutzen, müssen in das Verzeichnis „libraries\“, damit sie von der IDE korrekt als Beispiele erkannt werden.

Den Source-Code für die hier beschriebenen Experimente können Sie von [4] herunter laden.

(110661)

Weblinks

[1] http://chipkit.cc

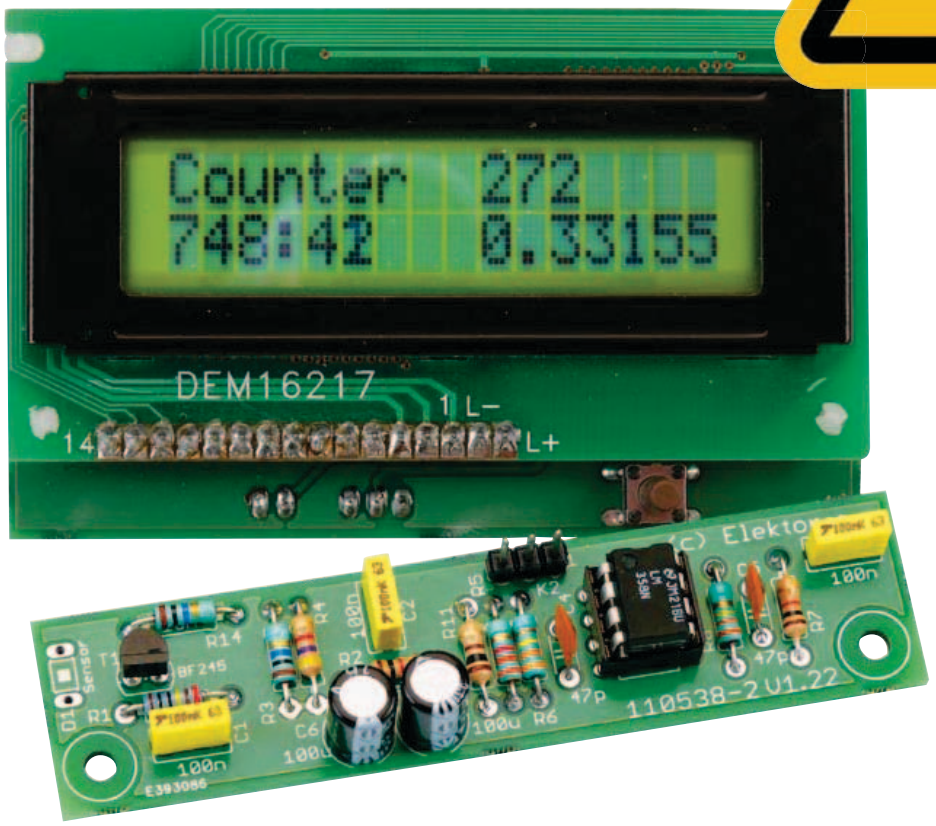
[2] www.digilentinc.com

[3] http://arduino.cc

[4] www.elektor.de/110661

Verbesserter Strahlungsmesser

Der Alpha-/Beta-/Gamma-Zähler



Von Burkhard Kainka (D)

Für die Messung radioaktiver Strahlung braucht man nicht viel mehr als eine PIN-Fotodiode und den passenden Sensorverstärker. Hier wird ein optimierter Vorverstärker mit einem Mikrocontroller-Zähler vorgestellt. Der Controller übernimmt auch gleich die Zeitmessung und zeigt die Impulsrate in „counts per minute“ an.

Das Gerät kann mit unterschiedlichen Sensoren für Gamma- und Alphastrahlung verwendet werden. Es eignet sich gut für Langzeitmessungen und für Untersuchungen an schwach strahlenden Proben. Im Vergleich zu einem Geiger-Zählrohr zeigt die Fotodiode auf Grund ihrer geringen Ausdehnung zwar eine geringere Nullrate, die Strahlung kleiner Proben hebt sich dafür aber umso besser vom Untergrund ab. Ein weiterer Vorteil des Halbleiter-Sensors ist die Möglichkeit, die Energie jedes einzelnen Teilchens zu messen. Damit können Proben genauer untersucht werden als mit einem Geigerzähler. Mit einer optionalen PC-Software erfasst man das Energiespektrum und kann daraus Rückschlüsse auf das Untersuchungsobjekt ziehen.

Vorverstärker

Die in Elektor 6/2011 vorgestellten Versuche mit einer Fotodiode BPW34 als Gamma-Detektor [1] waren nicht ganz einfach, weil sie extrem kurze Impulse lieferten. Nun soll ein optimierter Verstärker dafür sorgen, dass auch ohne einen Komparator direkt hörbare und auswertbare Impulse entstehen. Der Signalverstärker verwendet am Eingang einen JFET BF245B und einen nachfolgenden OPV-Verstärker mit insgesamt 30.000-facher Spannungsverstärkung. Am Ausgang findet man bis zu 200 mV hohe Impulse mit einer Breite von ca. 0,5 ms, die ohne weitere Verarbeitung hörbar gemacht werden können oder einen Zähler ansteuern können.

Die Schaltung (siehe **Bild 1**) kann mit mehreren parallel geschalteten Fotodioden betrieben werden. Damit erhöht sich die Impulsrate. Zugleich sinkt aber die Signalspannung, weil nun größere Kapazitäten im Spiel sind. Damit sinkt auch die Empfindlichkeit gegenüber schwachen Signalen, was den Vorteil teilweise wieder zunichte macht.

Elektor Produkte & Service

- Platine 110538-1
- Kit (Bauteile und Platine) 110538-71
- USB-FT232R-Breakout-Board 110553-91
- Kostenloser Layout-PDF-Download unter [2]
- Kostenloser Software- und Firmware-Download (Datei 110538-11 unter [2])

Eigenschaften

- Misst α, β- und γ-Strahlung
- Einfacher Aufbau mit Standard-Bauteilen
- Anschluss an den PC über Elektor-USB-FT232R-Breakout-Board
- Ansprechschwelle (Threshold) per Software einstellbar (via PC)
- Für zwei Sensortypen verwendbar

Der JFET am Eingang bringt einen guten Rauschabstand bei hohem Eingangswiderstand. Am Source-Widerstand des BF245B findet man eine Gleichspannung im Bereich 2 V bis 3 V, die weitgehend unabgängig von der Betriebsspannung ist (beim BF245C wäre es mehr, beim A-Typ weniger). Damit erreicht man einen passenden Arbeitspunkt für den Operationsverstärker. Die Fotodiode arbeitet mit der vollen Betriebsspannung, weil die Gate-Spannung über 20 MΩ auf Null gezogen wird. Dies ist wichtig, weil die Diodenkapazität mit steigender Spannung sinkt.

Der Zähler

Der Impulszähler wird hier mit einem ATmega88 und einem zweizeiligen LCD realisiert. Die Versorgungsspannung von 9...12 V gelangt über D1 als Verpolungsschutz zum Spannungsregler IC2 (78L05), der 5 V für den Mikrocontroller und den Sensor liefert. Die Platine besitzt einen ISP-

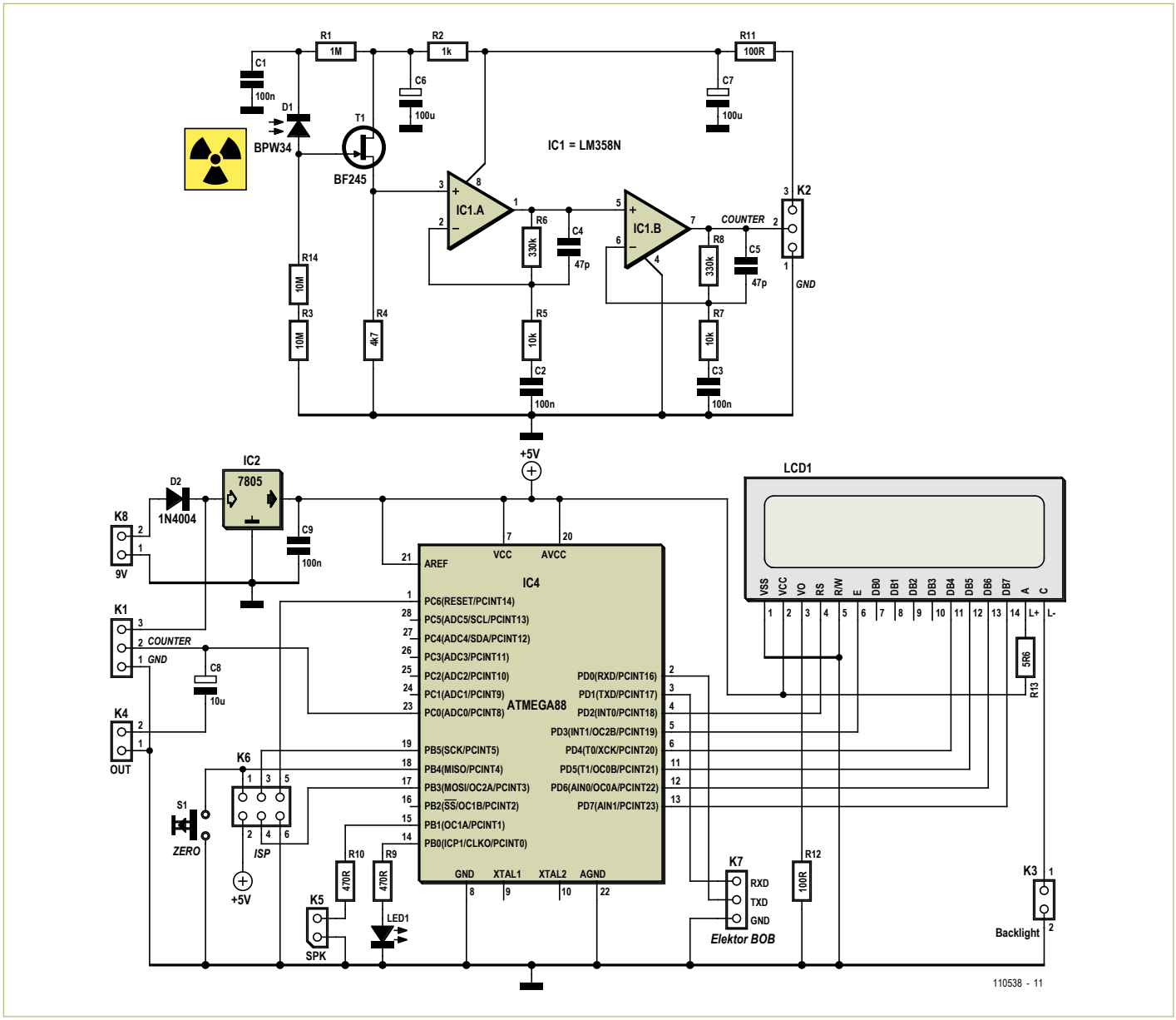


Bild 1. Schaltbild von Vorverstärker und Controllerplatine.

Stückliste

Widerstände:
R1 = 1 M
R2 = 1 k
R3,R14 = 10 M
R4 = 4k7
R5,R7 = 10 k
R6,R8 = 330 k
R9,R10 = 470 Ω
R11,R12 = 100 Ω
R13 = 5Ω6

Kondensatoren:
C1,C2,C3,C9 = 100 n
C4,C5 = 47 p
C6,C7 = 100 µ/ 16V
C8 = 10 µ/ 16V

Halbleiter:
D1 = BPW34
D2 = 1N4001
D3 = 5-mm-LED, grün
IC1 = ATmega88PA-PU (Atmel), programmiert
IC2 = LM358N
IC3 = 78L05
T1 = BF245B

Außerdem:
S1 = 1-poliger Taster
K1…K8 = Stiftleiste, z.B. TE-Connectivity 3-826926-6
LCD1 = DEM16217, bei Elektor erhältlich (030451-72)

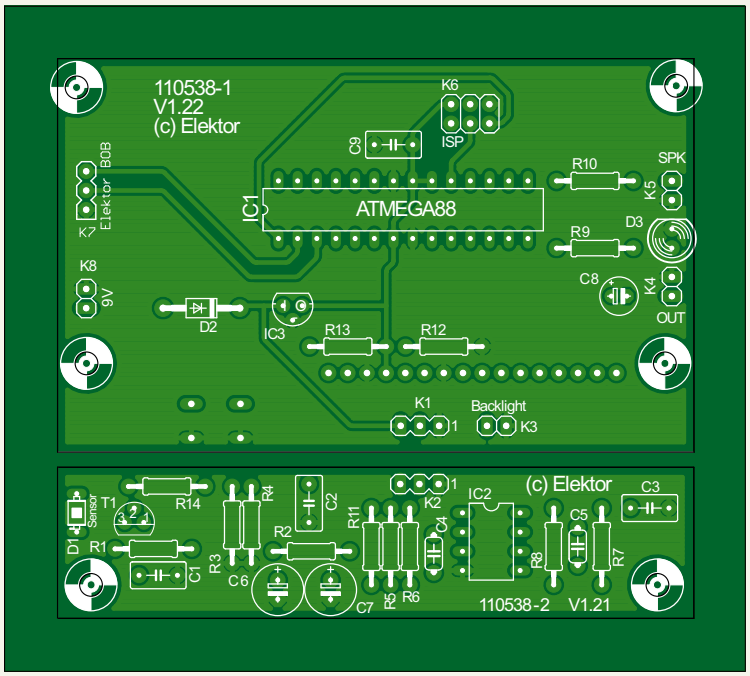


Bild 2. Die Platinen des Strahlungsmessers.

Anschluss für bequeme Software-Updates und hat zusätzlich einen Anschluss für die serielle Schnittstelle (K1), über die man Daten zum PC übertragen oder Einstellungen vom PC lesen kann. Die Signale RXD und TXD haben TTL-Pegel und passen für den Anschluss des USB-FT232R-Breakout-Boards aus Elektor 9/2011.
Der Anschluss K1 stellt die Verbindung zum Sensorverstärker her. Der Signalausgang mit seiner Ruhe-Gleichspannung und den überlagerten Nutzsignalen gelangt direkt an den Analogeingang ADC0 des Controllers. Beim Start muss die Controllersoftware den mittleren Ruhepegel bestimmen. Alle um einen definierten Level darüber hinausgehenden Impulse werden gezählt. Der Schalter S1 dient zum Start einer neuen Messung, ohne dass der Ruhepegel neu bestimmt werden muss. Damit hat man die Chance, den Nullpegel zuerst ohne Messobjekt zu ermitteln und dann erst mit dem Messobjekt zusammen die eigentliche Messung zu starten.
Alle gezählten Impulse erzeugen zugleich ein Ausgangssignal für die LED und am Ausgang K5, wo ein kleiner Lautsprecher angeschlossen werden kann. Der Zählvorgang kann so optisch und akustisch verfolgt werden. Der Lautsprecher mit 8 Ω bis 32 Ω kann auch über einen Lautstärkesteller (logarithmisches Poti mit 1 kΩ) angeschlossen werden, denn bei Langzeitmessungen kann das Ticken des Zählers auch mal nerven.
Das unverarbeitete Sensorsignal wird zusätzlich über C8 an den Ausgang K4 gelegt und damit zum Beispiel über eine BNC-Buchse ausgekoppelt, an die man beispielsweise ein Oszilloskop anschließen kann. Schließt man hier einen Audioverstärker an, ist das Ticken der Strahlung zu hören. Man kann sogar die unterschiedlichen Energien der einzelnen Teilchen heraushören.

Platine

Das Projekt verwendet eine Platine mit zwei Sektionen (**Bild 2**). Die Sensorplatine ist abtrennbar und kann über ein dreipoliges Kabel mit der Zählerplatine verbunden werden. Damit hat man die Möglichkeit, den Sensor in ein lichtdichtes Gehäuse einzubauen.
Das LCD und der Tastschalter müssen auf der Rückseite montiert werden, alle anderen Bauteile auf die Vorderseite. Beim Strahlungssensor D1 hat man die Wahl und kann überlegen, wie das Gerät später eingesetzt werden soll.
Testen Sie die Schaltung am besten zuerst einmal ohne die Sensordiode. Der OPV muss eine mittlere Gleichspannung an seinem Ausgang zeigen. Und der Eingang muss so empfindlich sein, dass eine Annäherung mit dem Finger bereits ein Signal liefert, das vom Zähler erkannt wird.
Der Sensor kann ebenfalls auf der Rückseite bestückt werden, wie in **Bild 3** zu sehen ist. Die sensible Fläche des Messgeräts liegt dann unten rechts auf der Platine. Wichtig ist, dass man die Fotodiode BPW34 lichtdicht einbaut und den Bereich um die Diode sorgfältig abschirmt. Beim Einbau sollte ein Stückchen schwarzes Isolierband unter die Fotodiode montiert werden. Es soll verhindern, dass später Licht durch die grünlichtransparente Platine eindringen kann. Die Bauteilseite der bestückten Musterplatine ist in **Bild 4** zu sehen.
Die Platine muss auf beiden Seiten im Bereich um die Fotodiode mit Alufolie abgedeckt werden, die zugleich leitend mit Masse verbunden wird. Nur so erreicht man eine gute Abschirmung gegen Licht und gegen Störfelder, die zu falschen Signalen führen könnten. Unter der Alufolie soll weiteres Isolierband dafür sorgen, dass kein Kurzschluss mit Teilen der Schaltung entsteht. Die Kontaktierung der Alufolie erreicht man zum Beispiel mit einer Schraube und zwei Unterlegscheiben. Die

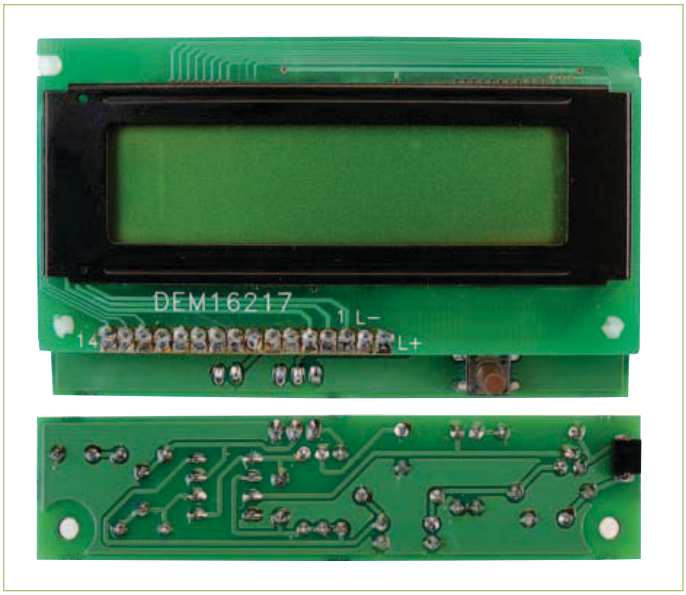


Bild 3. Bestückung auf der Platinenrückseite.

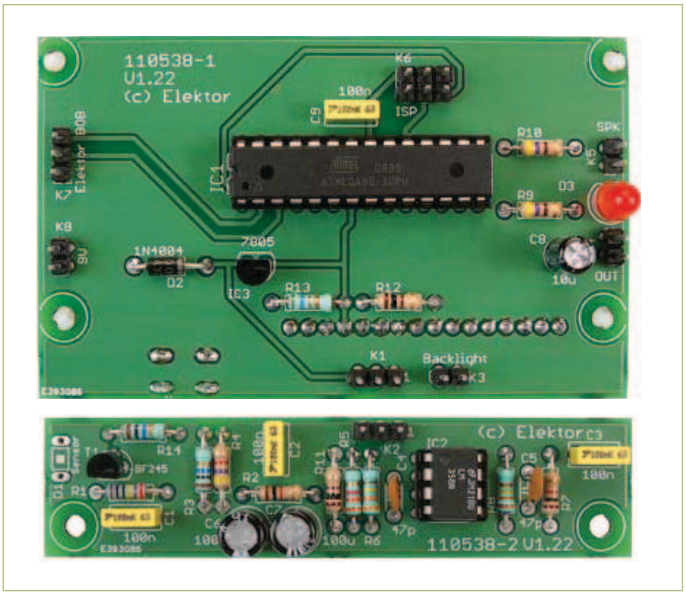


Bild 4. Die Bestückungsseite.

Alufolie soll über der Fotodiode eng aufliegen. Hat sie in diesem Bereich zu viel Abstand, baut man ungewollt ein Kondensatormikrofon, der Zähler kann dann auf lauten Schall reagieren.
Testen Sie die Funktion der Platine nun mit der lichtdicht eingebauten Fotodiode. Am Ausgang erscheint wieder die Gleichspannung von ca. 2 V bis 3 V. Man kann damit kontrollieren, ob die Schaltung wirklich lichtdicht verpackt ist. Falls der Arbeitspunkt sich nach oben verschoben hat, dringt wahrscheinlich Licht ein. Wenn alles in Ordnung ist, zeigt das Oszilloskop nur ein gleichmäßiges Rauschen in der Größenordnung von ca. 5 mV_{SS}.
Jetzt kann man das Gerät mit einer radioaktiven Probe testen. Legen Sie zum Beispiel ein radioaktives Mineral direkt auf den Sensor. Die von der Probe ausgehenden Gammastrahlen erzeugen Signale, die sich deutlich aus dem Rauschen abheben. Jeder Impuls über einem bestimmten Niveau wird gezählt. Dieser Triggerlevel kann später per Software justiert werden. Falls Sie keine radioaktive Probe zur Hand haben, heißt es warten. Spätestens nach ein paar Minuten wird ein Teilchen der Höhenstrahlung Ihren Sensor treffen und gezählt werden (**Bild 5, 6 und 7**).

Alpha-Messungen

Eine BPW34 verwendet eine Plastik-Umhüllung, die zu dick ist, um Alphastrahlung passieren zu lassen. Statt einer BPW34 können Sie aber auch eine BPX61 einsetzen, bei der das Glasfenster entfernt wurde. Die eigentliche Fotodiode liegt dann völlig frei und ist empfindlich für Alpha-Teilchen. Diese erzeugen im Vergleich zu Gammastrahlen etwa zehnmal größere Signale. Sie können den gleichen Sensorverstärker einsetzen, der aber nun abgetrennt in einem eigenen, lichtdichten und abgeschirmten Gehäuse betrieben werden muss. Die zu untersuchende Probe muss mit in diese Dunkelkammer, denn selbst eine Alufolie wäre zu dick für die Alphastrahlung.
Zum Entfernen der Glasscheibe der BPX61 hat sich ein kleiner, schnell rotierender Schleifstein (Dremel) bewährt. Gehen Sie äußerst vorsichtig ans Werk, denn eine Beschädigung des

Strahlung messen mit erschwinglicher Photodiode

Bild 5. Messung der Nullrate: 0,33 Impulse/ Minute sind normal.

Bild 6. Ein einzelner Messimpuls.

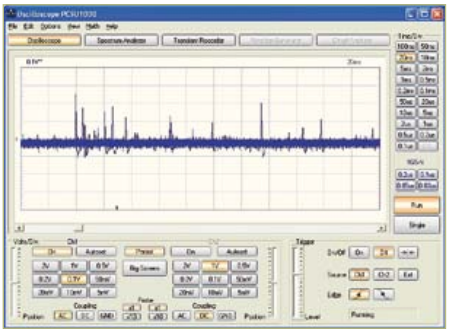


Bild 7. Hintergrundrauschen und Nutzsignale.

Listing 1:
Mittelwert und Triggerlevel

```
Readeeprom L , 1
 If L = 255 Then L = 10
 U = 0
 For N = 1 To 1000
   D = Getadc(0)
   U = U + D
 Next N
 U = U / 1000
 Um = U
 U0 = Um + L
 N = 0
```

Listung 2:
Erfassung eines Impulses

```
Do
   Max = U0
   Do
     D = Getadc(0)
   Loop Until D > U0
   Portb.0 = 1
   Portb.1 = 1
   If D > Max Then Max = D
   Do
     D = Getadc(0)
     If D > Max Then Max = D
   Loop Until D < U0
   N = N + 1
   Max = Max - Um
   If Max > 255 Then Max =
255
   Print Chr(max);
   Portb.0 = 0
   Portb.1 = 0
 Loop
```

Listung 3:
Zeiterfassung und LCD-Ausgaben

```
Tim1_isr:
  Timer1 = -7812
  S = S + 1
  If S = 60 Then
    S = 0
    M = M + 1
    Count = N
    Count = Count / M
    Locate 2 , 10
    Lcd Count
    Lcd „     „
  End If
  Locate 1 , 10
  Lcd N
  Locate 2 , 1
  Lcd M
  Lcd ":"
  Lcd S
  Lcd "  "
```

Listing 4:
Auswertung der gesendeten Energiewerte

```
Private Sub Timer1_Timer()
  While INBUFFER() > 0
    d = READBYTE()
    bin(d) = bin(d) + 1
  Wend
  For n = 1 To 255
    x1 = 2 * n
    x2 = 2 * n + 2
    y1 = 200 - bin(n)
    y2 = 200 - bin(n + 1)
    If y1 > 255 Then y1 = 255
    If y2 > 255 Then y2 = 255
    Picture1.Line (x1, y1)-
(x2, y2)
  Next n
End Sub
```

Listung 5:
Einstellung der Triggerschwelle

```
Private Sub Command2_Click()
  l = HScroll1.Value
  SENDBYTE l
End Sub

Private Sub Command4_Click()
  l = 100 + HScroll1.Value
  SENDBYTE l
End Sub
```

Diodenkristalls oder des Bonding-Drähtchens wäre das Ende der Diode.
Als Abschirmung eignet sich eine Blechdose (siehe **Bild 8**). Das Blech muss unbedingt mit an die Signalmasse angeschlossen werden, damit es gleichzeitig als elektrische Abschirmung wirkt. Erst wenn der Deckel aufgesetzt ist, kann die Messung beginnen. Auf dem Oszilloskop sieht man die großen Alpha-Signale mit Spitzen bis zu 2 V. Gleichzeitig erkennt man aber auch schwächere Signale, denn die BPX61 ist genauso wie die BPW34 für Gammastrahlen empfindlich. Die Strahlenart kann deshalb an der Impulshöhe erkannt werden.

Firmware

Die Firmware (kostenloser Download unter [2]) wurde in Bascom-AVR geschrieben und ist relativ einfach und leicht überschaubar.

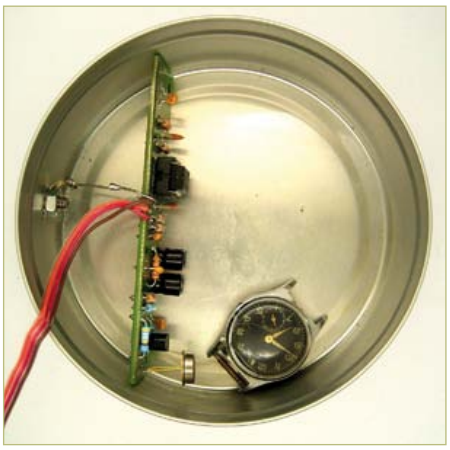


Bild 8. Abschirmung.

Die Ausgangsspannung des Vorverstärkers liegt im Ruhezustand bei ca. 2 V. Dazu kommen dann im Betrieb die Sensorimpulse. Damit ein Zähler diese Impulse verwerten kann, müsste eigentlich ein Komparator eingesetzt werden. Doch der ATmega ist schnell genug, um diese Aufgabe selbst zu erledigen. Dazu wird beim Start eine Nullmessung mit Mittelung über 1000 Messpunkte durchgeführt (siehe **Listing 1**). Der gewonnene Mittelwert U wird um den Triggerlevel L erhöht, um einen genügenden Abstand zum Rauschen zu bekommen und dient dann als Vergleichswert U_0 zur Auswertung der Zählimpulse.
Während der eigentlichen Messung (**Listing 2**) steuert der Software-Komparator auch die beiden digitalen Ausgänge PortB.0 und PortB.1 an. An B.0 ist eine LED angeschlossen, die für jeden erkannten Impuls einen Lichtblitz abgibt. An B.1 kann ein Minilautsprecher mit Vorwiderstand (wahlweise auch mit einem Lautstärkepoti) angeschlossen werden.
Zusätzlich wird das Maximum eines jeden Impulses gesucht und als ein Byte über die serielle Schnittstelle an den PC gesandt. Nur ein Byte deshalb, weil damit kein Zeitverlust auftritt. Zugleich bedeutet das, dass die Impulshöhe auf den Wert 255, also auf 1,25 V beschränkt ist. Größere Impulse können vorkommen, werden aber als 255 gemeldet.
Das Display zeigt laufend den aktuellen Zählerstand, wobei die Anzeige einmal pro Sekunde aktualisiert wird (**Listing 3**).

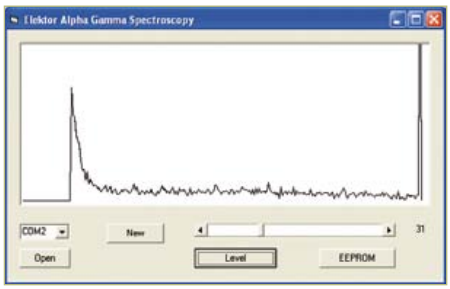


Bild 9. Das Alpha-Spektrum einer Pechblenden-Probe.

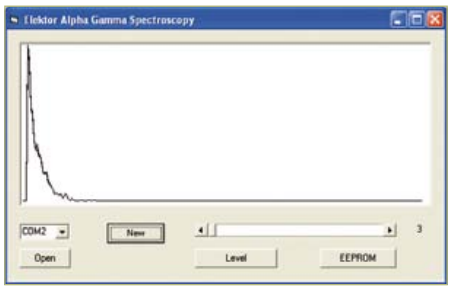


Bild 10. Gammastrahlung: Pechblende hinter einer Alufolie.

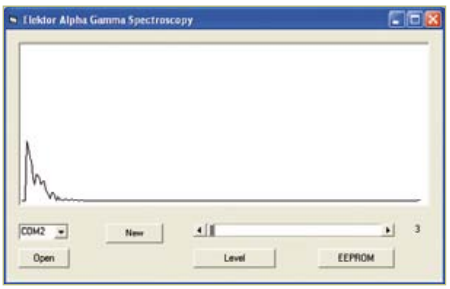


Bild 11. Beta-Messung an Kaliumchlorid.

Außerdem wird in der unteren Zeile die Messzeit in Minuten:Sekunden seit dem Start der Messung angezeigt. Zusätzlich berechnet das Programm nach jeder vollen Minute die Zählrate in Impulsen pro Minute. Die Empfindlichkeit des Messgeräts hängt in großem Maße von der Auslöseschwelle des Komparators ab. Diese ist auf den Wert 10 voreingestellt, kann aber nachträglich über die Schnittstelle verändert werden.

Wegen der relativ dicken Probe verlieren die Alpha-Teilchen unterschiedlich viel Energie auf dem Weg zum Sensor. Man sieht daher an dieser Probe keine scharfen Linien. Bei reinen Gamma-Messungen mit der BPW34 bleibt der obere Bereich leer. Alpha-Strahlen enthalten dagegen oft noch Energien außerhalb des Anzeigebereichs, die als spitze Nadel beim Maximalwert zusammengefasst werden.

enthaltene Kalium-40 ist radioaktiv. 90 % der Zerfälle erzeugen ein Beta-Teilchen mit einer maximalen Energie von 1,3 MeV. Die restlichen 10 % erzeugen ein Gamma-Quant mit 1,5 MeV. Das Beta-Spektrum zeigt eine charakteristische, abfallende Energieverteilung mit einer klaren maximalen Energie. Das Gamma-Spektrum zeigt eher eine scharfe Linie. Das Gesamtspektrum (**Bild 11**) hat den erwarteten Verlauf. Damit zeigt sich, dass die Fotodiode prinzipiell Alpha-, Beta- und Gamma-Strahlung erfasst.

(110538)

Hintergrund-Strahlung messen

Eine Schwelle von drei A/D-Stufen hat sich als günstig erwiesen, weil damit auch Impulse erfasst werden, die gerade noch aus dem Rauschen ragen. Um die Schwelle zu ändern, muss nur ein einziges Byte an das Gerät gesandt werden. Werte bis 100 werden direkt als neue Auslöseschwelle übernommen. Will man eine neue Schwelle als neue Standardeinstellung im EEPROM des Controllers speichern, addiert man 100 hinzu. Ein Byte 103 wirkt sich also nicht direkt aus, sondern legt die Schwelle beim Neustart auf 3 fest.

PC-Software

Das VB-Programm AlphaGamma (kostenloser Download unter [2]) empfängt alle ankommenden Bytes und ordnet sie in die 255 zugehörigen Speicher (Bins) ein. Nach einiger Zeit zeigt sich dann, bei welcher Energiestufe besonders viele Ereignisse gemessen wurden. Das Energiespektrum wird in einem einfachen Diagramm dargestellt (**Bild 9**). Die höheren Energien gehören grundsätzlich zu Alpha-Teilchen.

Listung 4 zeigt die entscheidende Timer-Routine. Hier werden jeweils alle Bytes gelesen und ausgewertet, die sich gerade im Puffer befinden. Mit den gelesenen Werten wird das Array-Bin(255) gefüllt und grafisch dargestellt.

Das Programm erlaubt zugleich auch die Einstellung der Auslöseschwelle im Bereich 2 bis 100. Wahlweise kann die Einstellung direkt übernommen oder ins EEPROM gespeichert werden (**Listing 5**).

Alpha-Teilchen lassen sich leicht abschirmen. Bei der offenen BPX61 reicht dazu bereits eine Lage Alufolie aus. Das Spektrum der Pechblende verschiebt sich damit deutlich nach unten (**Bild 10**).

Auch Beta-Teilchen sollten messbare Signale zeigen, die allerdings ähnliche Impulshöhen aufweisen wie Gammastrahlen und daher nur schwer zu unterscheiden sind. Als Test für die Beta-Empfindlichkeit wurde eine Langzeitmessung mit der BPX61 an einer kleinen Probe Kaliumchlorid durchgeführt. Das darin zu einem kleinen Prozentsatz

Weblinks:

[1] www.elektor.de/110372

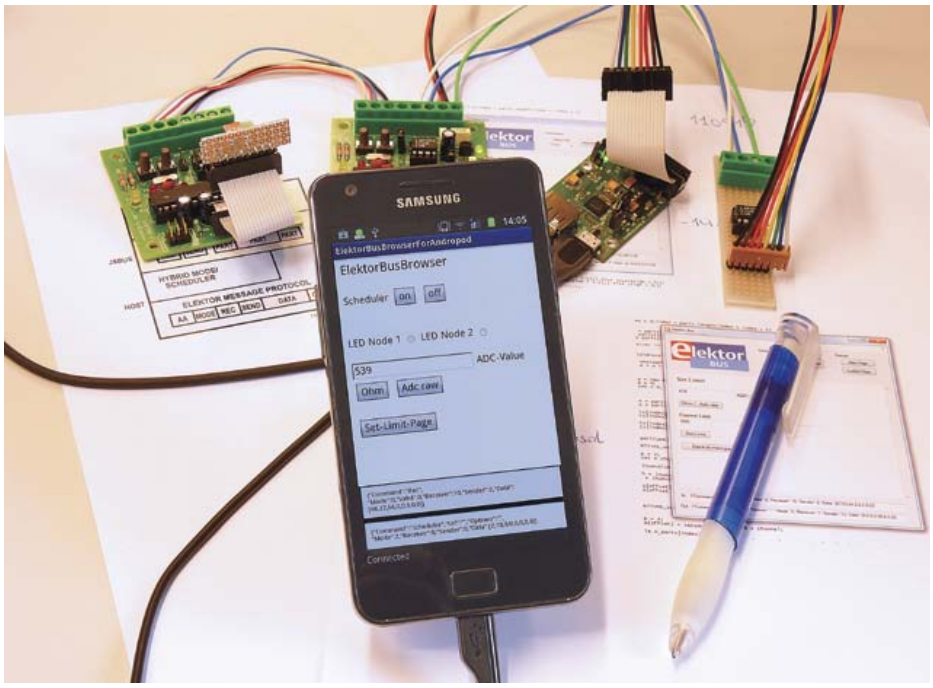
[2] www.elektor.de/110538

Bauteilsatz

Zu diesem Projekt können Sie einen Bausatz erwerben, der die Platine und alle Bauteile (inklusive programmiertem Controller) enthält. Ein passendes Display kann optional mitbestellt werden. Den Preis und weitere Informationen findet man auf der Website zu diesem Artikel [2].

Hier kommt der Bus (9)
Schnell zur eigenen Steuerung

Wer eine eigene Bus-Anwendung ausprobieren wollte, musste sich bisher mit kleinen Änderungen unserer Demos zufrieden geben - oder alles neu programmieren. Das wird nun anders: Die eigene Bus-Steuerung und ein maßgeschneidertes User-Interface sind rasch erstellt und bei Bedarf im Nu erweitert. Und da das Konzept auf HTML und Javascript basiert, kann dieselbe Zentrale auf so unterschiedlichen Plattformen wie PC und Smartphone zum Einsatz kommen.



ElektorBusBrowser auf einem Android-Smartphone.

Von Jens Nickel

Die Domotik und andere Anwendungen aus dem Bereich Messen, Steuern und Regeln verlangen nach einer Schaltzentrale mit Display, auf der Werte angezeigt werden und Anwender Einstellungen ändern können. So eine Zentrale kann zum Beispiel ein PC sein, der in VisualBasic programmiert wird. Aber auch ein Smartphone oder Tablet, auf dem das Open-source-Betriebssystem Android läuft, gibt eine prima Steuerungszentrale ab. Eigene Anwendungen strickt man dort mit der Sprache Java und dem leistungsfähigen Android-Framework, das für Einsteiger allerdings eine recht steile Lernkurve bereithält. Wer Erfahrung in der jeweiligen Sprache, dem Framework und dazu noch den jeweiligen Entwicklungstools mitbringt, kann die ElektorBus-Protokolle samt der eigenen Steuerungs-Applikation natürlich selbst implementieren. Wenn sich der Code für die Protokolle und die eigentliche Anwendung vermischen (so wie in der bisher vorgestellten Demo-Software), sind Änderungen und Erweiterungen allerdings schwer einzubringen. Und wechselt man von einer Geräte-Plattform zur anderen, muss man wieder von Null anfangen.

Daher benötigen wir eine Bibliothek, die

- die ElektorBus-Protokolle bereits implementiert, so dass sich der Entwickler ganz auf die eigene Bus-Applikation konzentrieren kann
- Anwendungscode klar vom Protokollcode trennt
- eine einfache Programmierung und Gestaltung von Benutzeroberflächen gestattet, was vielen Elektronik-Praktikern entgegenkommen dürfte
- noch dazu plattformunabhängig ist, so dass dieselbe Anwendung sowohl auf einem PC als auch auf einem Smartphone laufen kann.

Gibt's nicht? Doch! Hier kommt sie!

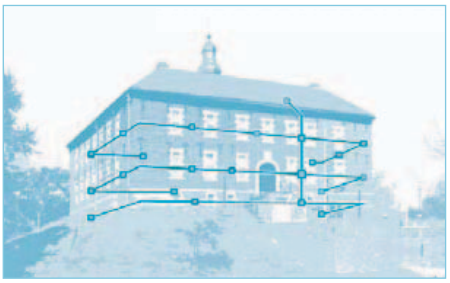
HTML versteht man überall

Bevor wir dazu kommen, wie die Bibliothek genutzt wird, wollen wir uns erst einmal ansehen, wie das Ganze funktioniert. Auf den ersten Blick mag das Konzept recht komplex und vielleicht sogar umständlich anmuten. Doch sind die Vorteile gegenüber einer herkömmlichen Programmierung in der Praxis ziemlich eindrucksvoll, so dass sich die Idee auch für viele andere Elektor-Projekte eignen dürfte, die eine PC-

Elektor Produkte & Service

- Experimental-Knoten (Platine 110258-1 oder 3er-Set Platinen 110258-1C3)
- USB/RS485-Konverter (fertig aufgebaut und getestet 110258-91)
- Gratis Software-Download (Controller-Firmware plus PC-Software)

Alle Produkte und Downloads sind über die Website zu diesem Artikel erhältlich: www.elektor.de/110517



Steuerung benötigen. Außerdem werden ähnliche Systeme mit Erfolg in der modernen Softwareentwicklung eingesetzt, zum Beispiel bei mobilen „Apps“. Daher wollen wir auch Einsteigern empfehlen, sich einmal durch die folgenden Abschnitte durchzubeißen.

Zuerst zur Plattformunabhängigkeit: Diese erreichen wir, indem die eigentliche Bus-Anwendung samt Benutzeroberfläche in HTML und Javascript programmiert wird. Dieses Duo ist ein wahrer Tausendsassa, entsprechende UIs können (im Browser) auf einem Windows-PC, einem Mac, einem Linux-Rechner und diversen Mobilgeräten dargestellt werden. Als weiterer Vorteil ergibt sich, dass HTML-Formulare über das Internet transportiert werden können, was uns später faszinierende Möglichkeiten der Fernsteuerung eröffnen wird. Darüber hinaus befindet sich HTML auf dem aufsteigenden Ast: HTML5 macht viele neue Features wie zum Beispiel lokale Datenbanken, 3D-Grafik und mehr möglich.

Spezial-Browser

Die später erstellten HTML-Oberflächen für unseren Bus sind zwar in einem „normalen“ Web-Browser wie Firefox oder Internet Explorer darstellbar, jedoch können diese Browser aus Sicherheitsgründen viel weniger als eine normale Anwendung. Ein Standard-Browser ist zum Beispiel nicht dazu zu bewegen, Daten über eine serielle Schnittstelle zu empfangen oder zu versenden. Daher backen wir uns einen Spezial-Browser, der ebenfalls HTML-Formulare darstellen kann, aber auf den ElektorBus zugeschnitten ist. Da der *ElektorBusBrowser* auf den USB und andere Gerätefunktionen zugreift, muss es für jede Plattform einen eigenen geben. Das stört aber nicht weiter, da der Quellcode vom Anwender normalerweise nicht verändert werden muss. Daher werden wir den *ElektorBusBrowser* auch als fertige Anwendungsdatei anbieten (zum Beispiel als .exe für den PC). Wenn man von einem Gerät auf ein anderes wechselt, muss man nur den entsprechenden *ElektorBusBrowser* installieren und die HTML/Javascript-Dateien der eigenen Anwendung in einem bestimmten Ordner ablegen. Fertig!

Der Screenshot in **Bild 1** zeigt den ersten realisierten *ElektorBusBrowser*: Wie bei einem normalen Browser nimmt das Fenster, das die HTML-Inhalte der eigentlichen Anwendung darstellt, den größten Raum ein. HTML und Javascript formen sozusagen das Innere, der in einer herkömmlichen Programmiersprache wie VB.NET oder Android-Java programmierte Browser das Äußere der Bus-Anwendung (siehe **Bild 2**). Den *ElektorBusBrowser* kann man auch als *Host* („Gastgeber“) bezeichnen.

Protokoll-Bibliothek

Dieser *Host* nimmt eine *ElektorBusMessage* über die serielle Schnittstelle des jeweiligen Geräts entgegen. Die Message wird dann gemäß den bisher beschriebenen Protokollen verarbeitet; die Nutzdaten werden daraufhin an die eigentliche Steuerungs-Anwendung (HTML und Javascript) weitergereicht. Drückt der User einen Button in der HTML-Benutzeroberfläche, dann generiert Javascript die entsprechenden Nutzdaten der zu sendenden Message (ein Beispiel aus dem letzten Teil könnte das Setzen des Grenzwertes auf dem Sensor sein). Die Informationen werden nun „nach außen“ an den *ElektorBusBrowser* weitergereicht, der die Message verschickt.

Prinzipiell hätte man alle drei aufeinander aufbauenden Bus-Protokolle *ElektorMessageProtocol*, *HybridMode* (optional) und *ApplicationProtocol* innerhalb des Hosts implementieren können. Auf der anderen Seite wäre es möglich gewesen, alle 16 empfangenen Roh-Bytes nach „innen“ weiterzureichen und die Protokolle vollständig mit Javascript abzuhandeln. Hier gehen wir jedoch einen gemischten Weg: das simple *ElektorMessageProtocol* und der etwas Timing-kritische *HybridMode* samt Scheduler werden innerhalb des Host-Codes abgehandelt; das *ApplicationProtocol*, das etwas mehr Code erfordert und von einigen Lesern vielleicht noch erweitert werden wird, wurde mit Hilfe einer kleinen Javascript-Bibliothek implementiert (**Bild 3**).Und so läuft das Ganze im Detail ab: Der *Host* empfängt die 16 Bytes einer über den Bus geschickten *ElektorBusMessage* mit Hilfe der in [1] beschriebenen Startbyte-Synchronisation. Die Nachricht wird daraufhin „entpackt“, wobei eine Daten-Struktur aus (unter anderem) der Senderadresse, Receiveradresse und den acht Nutzdaten-Bytes entsteht.

Bild 1. Screenshot des ersten *ElektorBusBrowsers* mit laufender Beispielanwendung.

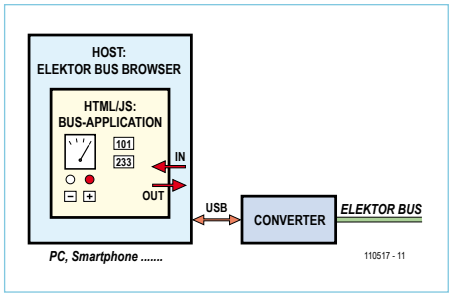


Bild 2. Der plattformabhängige Host ist nur für die Basisfunktionen (serielle Schnittstelle, Synchronisation) zuständig. Die eigentliche Bus-Anwendung ist in HTML/Javascript implementiert.

Diese Einzelteile werden zu einem einzigen String (*InCommand*) codiert und nach innen zu Javascript weitergereicht (siehe **Bild 4**). Das InCommand (siehe Kasten) ist reiner Text, was eine Tauglichkeit für die verschiedensten Plattformen sicherstellt.

Javascript nimmt das InCommand entgegen und wandelt den String in eine simple Datenstruktur namens *Message* zurück, die nun weiter decodiert werden kann. Gemäß dem *ElektorApplicationProtocol* entstehen sogenannte *Parts*, was man mit Informationseinheiten übersetzen könnte (siehe Kasten „Messages und Parts“). So eine Ein-

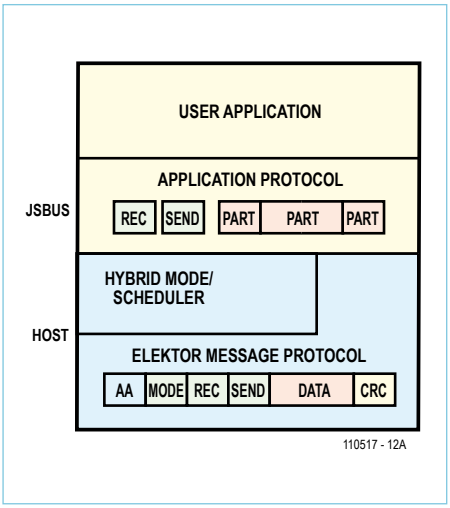


Bild 3. Standard-Protokoll-Stack für den ElektorBus (*HybridMode* und Scheduler sind optional). Für das *ElektorMessageProtocol* ist der Host, für das *ApplicationProtocol* die Javascript-Bibliothek JSBus zuständig.

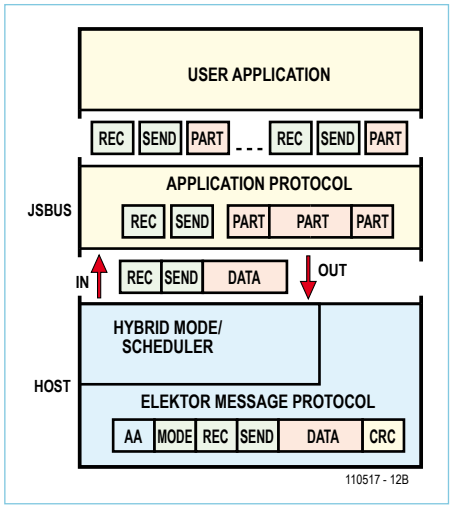


Bild 4. Der Host und die Library JSBus unterhalten sich, indem sie die Bestandteile der empfangenen und zu sendenden Nachrichten (in Text-Form) austauschen.

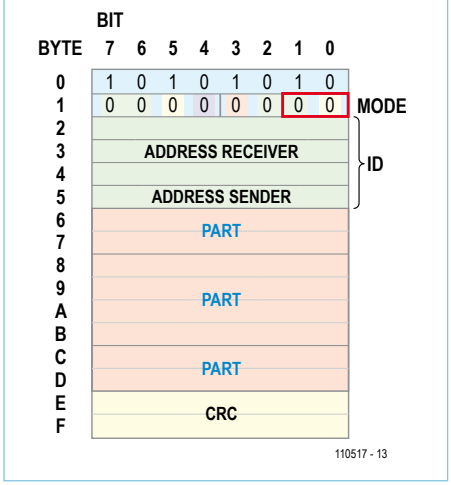


Bild 5. Das *ApplicationProtocol* erlaubt es, mit Hilfe der acht Datenbytes mehrere Zahlen-Werte oder andere Informationseinheiten (zum Beispiel Grenzwert-Alarme) gleichzeitig zu transportieren. Das Diagramm zeigt beispielhaft drei solcher *Parts*: zwei Mal mit zwei Byte und einmal mit vier Byte Länge.

heit kann zum Beispiel ein übertragener 2-Byte-Wert (-1023 bis 1023) sein oder die Meldung „Grenzwert von Sensor2 unterschritten!“. Da wir innerhalb einer Message mehrere solcher Infohäppchen transportieren können (**Bild 5**), erzeugt die Javascript-Bibliothek aus einer *Message* gleich mehrere *Parts* (ein Array von Parts). Die empfangenen Parts werden dann an eine Javascript-Funktion übergeben, die der Anwender mit eigenem Code füllen kann. Solcher Anwendungscode trägt dann zum Beispiel in eine HTML-Textbox den numerischen Wert ein, der mit dem Part übertragen wurde.

Kommando-Typen

Beim Senden läuft alles in die andere Richtung. Ein Klick auf einen HTML-Button ruft eine vom Anwender geschriebene Javascript-Funktion auf, die einen (oder mehrere) Parts generiert. Parts, die für den gleichen Empfänger bestimmt sind, lassen sich zu einer einzigen Message encodieren. Das Message-Objekt wird dann mit Hilfe eines *OutCommands* (wieder ganz normaler Text) von Javascript nach außen zum *ElektorBusBrowser* weitergereicht. Dieser generiert dar-

InCommand und OutCommand

Die in Javascript geschriebene Bus-Anwendung und der (zum Beispiel in VB.NET codierte) *ElektorBusBrowser* als Host kommunizieren miteinander über einen einfachen Text-String. Mit Hilfe der so genannten JSON-Syntax ist hierin eine Datenstruktur mit den Informationen encodiert, welche Javascript zum Host (nach außen) beziehungsweise der Host (nach innen) zur Javascript-Anwendung übermitteln muss. Die Datenstruktur des *In-* und *OutCommands* ist recht ähnlich:

OutCommand:

Command	Der Typ des Commands („Send”, „Url”, „Scheduler”)
Url	Dateiname der zu ladenden HTML-Seite (nur bei „Url“)
Options	reserviert für spätere Zwecke
Mode	Modebyte der zu sendenden Nachricht (wichtig für Acknowledge-Mechanismus)
Receiver	Receiveradresse
Sender	Senderadresse
Data	Array aus acht Datenbytes (bzw. Adressen von bis zu acht ScheduledNodes)

InCommand:

Command	Der Typ des Commands („Rec”, „Status”)
Mode	Modebyte der empfangenen Nachricht (bzw. Status 2 = ok, -1 = Error)
Valid	Checksumme ok? (noch nicht implementiert)
Receiver	Receiveradresse
Sender	Senderadresse
Data	Array aus acht Datenbytes

Ein InCommand in JSON-Syntax lautet beispielsweise:

```
{"Command":"Rec","Mode":0,"Valid":0,"Sender":
2,"Receiver":10,"Data":[0,0,64,1,0,0,0,0]}
```

In unserem ersten *ElektorBusBrowser* werden die In- und OutCommands zu Debugging-Zwecken angezeigt (siehe Bild 1, unten).

aus 16 Bytes, die er dann über den Bus verschickt. Nach getaner Arbeit gibt der *Host* eine Erfolgsmeldung an Javascript zurück. Hierfür wird ebenfalls ein InCommand verwendet, das nun vom Typ „Status“ ist.
Es gibt auch weitere Typen von OutCommands (siehe Kasten); hiermit kann die HTML/Javascript-Anwendung den Host steuern. Das OutCommand „Url“ veranlasst den Host, eine neue HTML-Seite nachzuladen. So kann man sich eine Anwendung bauen, die verschiedene Formulare umfasst, die sich zum Beispiel über ein Menü anwählen lassen. Das OutCommand „Scheduler“ schaltet den Scheduler an und aus. Im Daten-Array wird dabei eine Adressliste für bis zu acht abgefragte Knoten mitgegeben.

Beispiel-Applikation

Am besten zeigen wir anhand eines Beispiels, wie man mit der Javascript-Bibliothek und dem *ElektorBusBrowser* arbeitet. Die Hardware können wir 1:1 aus dem letzten Teil übernehmen, es ist lediglich eine minimal geänderte Firmware auf beide Knoten aufzuspielen (BASCOM-File unter [2]). **Bild 6** zeigt nochmals unser Equipment: Der mit einem Foto-Widerstand ausgestattete Knoten 2 übermittelt laufend Werte an den Domotik-Master. Vom Master aus lassen sich auf dem Sensor die Einheit für die Messwerte umstellen sowie ein unterer Grenzwert setzen. Ist der Grenzwert unterschritten, meldet sich der Sensor mit einer Alarm-Meldung. Der Master sendet dann eine Nachricht an den Knoten 1, dass das dort angeschlossene Relais anziehen soll. Außerdem wird dem Sensor die Alarm-Meldung bestätigt.
Die erste Version eines *ElektorBusBrowsers* für den PC kann man von der Website zu diesem Artikel downloaden [2], genauso wie die Javascript-Bibliothek JSBus.txt und die Beispiel-Applikation, die aus zwei Webseiten namens „Index.htm“ und „Limit.htm“ besteht. Den Ordner UIBus, der die letztgenannten drei Files enthält, muss man sich auf den Desktop ziehen. Nach dem Start von ElektorBusBrowser.exe wird das erste HTML-Formular angezeigt (muss immer „Index.htm“ heißen). Wenn man sich mit dem seriellen Port verbunden und den Scheduler gestartet hat, werden – wie bei der PC-Demosoftware des letzten Teils

Messages und Parts

Innerhalb der Javascript-Bibliothek wird mit zwei Datenstrukturen gearbeitet, um empfangene und zu sendende *Messages* beziehungsweise *Parts* (übertragene Informationseinheiten wie 2-Byte-Werte, Alarm-Meldungen, zu setzende Messgrößen usw.) zu beschreiben.

Das **Message**-Objekt besteht im Wesentlichen aus den bekannten Bestandteilen einer ElektorBusMessage:

Mode	Modebyte
Receiver	Receiveradresse
Sender	Senderadresse
Data	Array aus acht Datenbytes
Valid	Checksumme ok? (noch nicht implementiert)

Innerhalb der acht Datenbytes können gemäß dem *ApplicationProtocol* bis zu vier Parts übertragen werden. Solch ein **Part** wird durch folgende Eigenschaften charakterisiert:

Valid	Checksumme ok? (noch nicht implementiert)
Sender	Senderadresse
Receiver	Receiveradresse
Channel	Channel-Nummer
Setflag	Soll- oder Messwert?
Ackflag	AcknowledgeMessage oder Originalnachricht (Flag auf Application-Ebene)
Mode	Modebyte der Nachricht (mit Acknowledge-Flags auf Message-Ebene)
Parttype	Typ des Parts, hierzu sind Konstanten definiert: PARTTYPE_VALUE2, PARTTYPE_VALUE4, PARTTYPE_VALUEFLOAT, PARTTYPE_LIMIT, PARTTYPE_SCALE.
Numvalue	übertragener numerischer Wert (z.B. -1023..1023 bei PARTTYPE_VALUE2)
Limit	0 = alles ok, 1 = Grenzwert unterschritten, 2 = Grenzwert überschritten
Quantity	Physikalische Größe (0..127, siehe [3])
Unit	Einheit (0..3, siehe [3])
Scale	Zehnerpotenz für Skalierung (-15..+15)
Interval	reserviert
Preset	reserviert
Options	reserviert

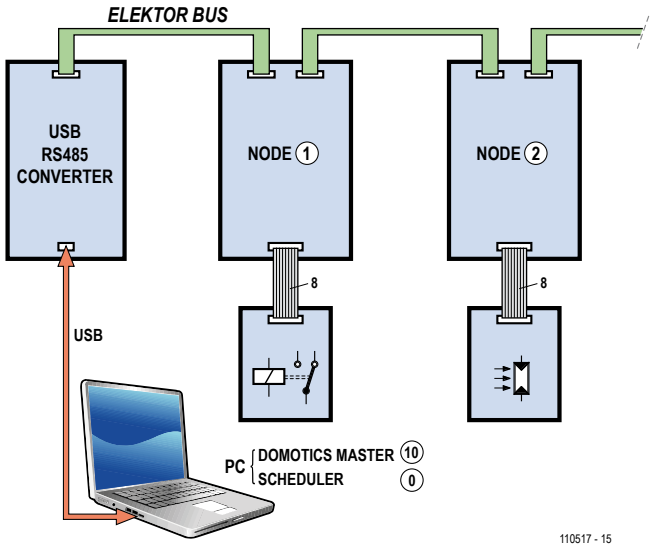


Bild 6. Für unser Beispiel nutzen wir die Hardware aus dem letzten Teil.

Bild 7. Grenzwert-Überwachung: Alles ist plattformunabhängig in HTML und Javascript realisiert.

- die Werte des Fotosensors in der Textbox angezeigt. Mit den darunter befindlichen Buttons kann man die Einheit zwischen ADC-Wert (0..1023) und Ohm umschalten. Die Alarm-Anwendung wurde dagegen auf ein zweites Formular ausgelagert, um den Einsatz des „Url"-Commands zu demonstrieren (**Bild 7**).

Reingeschaut

Nach diesem Test öffnen wir einmal die Datei „Index.htm" mit einem Texteditor, um den Sourcecode zu betrachten (siehe **Listing**). Ganz oben steht ein Verweis auf die Javascript-Library JSBus. Darunter folgt (durch ein zweites Paar Script-Tags eingeschlossen) der anwenderspezifische Javascript-Code. Die Funktion ProcessPart(part) {...} ist Pflicht; denn sie wird für jeden empfangenen Part einmal von JSBus aus aufgerufen. In der Funktion legt der Anwendungsentwickler fest, wie die empfangenen Werte, Alarm-Nachrichten und so weiter verarbeitet werden sollen. Auf die Eigenschaften (Properties) des Parts, zum Beispiel den Sender, den Channel und den übertragenen Wert greift man mit der Syntax `part.Eigenschaft` zu. Auch wer mit der C-ähnlichen Syntax von Javascript noch nicht vertraut ist, kann erkennen, dass wir überprüfen, von welchem Sender der jeweilige Part abgeschickt wurde. Auf Channel1 werden die LED-Werte übertragen (0 für Aus oder 1 für An). Wenn der Part zu Channel1 gehört, wird ein Radiobutton im HTML-Formular gesetzt. Dies wird mit der Funktion `RadioButtonSetvalue` erledigt, die in unserer JSBus-Library implementiert ist (zu den wichtigsten Funktionen siehe den Kasten). Als Parameter erwartet diese Funktion die ID des HTML-Radiobuttons sowie eine Zahl 0 oder 1 (Aus/An). Da `part.Numvalue` in unserem Fall gerade 0 oder 1 ist, können wir diesen Ausdruck direkt in den Funktionsaufruf einsetzen. Als IDs für die beiden Radiobuttons haben wir geschickterweise „LED1" und „LED2" gewählt. So können wir den ersten Parameter für die Funktion aus dem String „LED" und der Senderadresse zusammensetzen.

Bei einem Klick auf einen der HTML-Buttons wird die folgende Funktion `SetSensorScale(quantity)` aufgerufen. Nun muss ein Part gesendet werden, der den Fotosensor veranlasst, die folgenden Messwerte als ADC-Wert oder in Ohm zu übermitteln. Der Aufbau des entsprechenden Codes ist immer gleich: Mit der Zeile `var parts = InitParts();` initialisiert man zuerst ein leeres Array von Parts. Entscheidend ist die nächste Zeile: Dem bestehenden Parts-Array wird ein weiterer Part hinzugefügt. Die Bibliotheksfunktion `SetScale` erledigt dies einigermaßen komfortabel: Sie erwartet als Parameter nur die Sammlung von Parts, der ein weiteres Element hinzugefügt werden soll, die Sender- und Empfängeradresse, Channel und Modebyte, sowie die drei Kennwerte für das Setzen von Messgröße, Einheit und Skalen-Zehnerpotenz. Zurückgegeben wird das ergänzte Parts-Array, somit kann man dort mit weiteren Aufrufen der entsprechenden Javascript-Funktionen (siehe Kasten) weitere Parts anhängen.

Das Modebyte brauchen wir übrigens, damit wir einem Knoten, der in einer unsicheren *FreeBusPhase* sendet, eine gekennzeichnete *AcknowledgeMessage* senden können (siehe hierzu den Kasten „Ein neues Bit"). Im Beispiel teilt Knoten 1 den Status der eigenen Test-LED mit, wenn der Test-Button gedrückt wird. Das Versenden der entsprechenden Bestätigungs-Nachricht vom Master aus taucht im vorliegenden Anwendungscode aber nicht auf. Denn dies wird von der Javascript-Library automatisch erledigt (siehe die Funktion `ProcessReceivedParts(parts)` in JSBus.txt).

Rausgesendet

Mit `SendParts(parts, overrideQueue);` werden die erzeugten Parts (in unserem Fall nur ein einziges) encodiert und nach draußen geschickt. Die Library fasst dabei bis zu vier Parts, die im Parts-Array mit gleicher Sender- und Empfänger-Adresse hintereinanderstehen, automatisch zu einer einzigen Message zusammen. Parts, die nicht in eine einzige Message passen, werden zu mehreren Nachrichten encodiert, die dann nacheinander versendet werden. Auch darum muss sich der Anwendungsentwickler nicht kümmern. Die Messages werden von der Library in eine *Queue* eingetragen; wenn Javascript vom Host mitgeteilt bekommt, dass die erste Message versendet worden ist, kommt die nächste an die Reihe und so fort. Nun wird der zweite Parameter `overrideQueue` verständlich: Ein erneuter Aufruf von SendParts überschreibt entweder die noch nicht versendeten Nachrichten oder verfällt, falls sich in der Queue noch Nachrichten befinden (in der Datei Limit.htm nutzen wir beide dieser Möglichkeiten beim Setzen und Resetten des Grenzwert-Alarms, wobei vom Master jeweils zwei Messages versandt werden müssen).

Die nächsten beiden Zeilen des Javascript-Codes ändern einfachen Text in unserem HTML-Formular. Damit dies funktioniert, muss der Text ebenfalls eine ID besitzen („unit"). Je nach dem Wert von `quantity` wird der Text in „Ohm" oder „ADC-Value" geändert. In der Library sind für die wichtigsten Messgrößen noch ein paar Konstanten definiert, `RESISTANCE` steht beispielsweise für den Wert 18. BASIC-affine Leser aufgepasst: Man beachte die doppelten Gleichheitszeichen im Vergleich und denke strengstens an einheitliche Groß-/Kleinschreibung im Javascript-Code!

Nun zum HTML-Teil: Dieser besteht aus einer Reihe von Elementen, jeweils einge-

Listing: Die Datei Index.htm

```
<SCRIPT src='JSBus.txt' Language='javascript' ></SCRIPT>

<SCRIPT Language='javascript' >

function ProcessPart(part)
{
        if (((part.Sender == 1)||(part.Sender == 2)) && (part.Parttype == PARTTYPE_VALUE2))
        {
                if (part.Channel == 1) {RadioButtonSetvalue('LED' + part.Sender, part.Numvalue);};
        }

        if ((part.Sender == 2) && (part.Parttype == PARTTYPE_VALUE2))
        {
                if (part.Channel == 0) {TextboxSetvalue('ADC', part.Numvalue);};
        }
}


function SetSensorScale(quantity)
{
        var parts = InitParts();
        parts = SetScale(parts, 10, 2, 0, 0, quantity, 0, 0);
        SendParts(parts, true);

        if (quantity==RESISTANCE) {TextSetvalue('unit','Ohm');};
        if (quantity==RAWVALUE) {TextSetvalue('unit','ADC-Value');};
}

</SCRIPT>


<FORM Name='Bus'>

<STYLE type='text/css'>#head {font-size:20}</STYLE>

<DIV ID='head' >ElektorBusBrowser </DIV> <br/>

Scheduler
<BUTTON Type='button' onclick='javascript:SetScheduler(SCHEDULER_ON,2,10,0,0,0,0,0,0)' >on</BUTTON>
<BUTTON Type='button' onclick='javascript:SetScheduler(SCHEDULER_OFF,2,10,0,0,0,0,0,0)' >off</BUTTON>
<br/><br/><br/>

LED Node 1
<INPUT Type='radio' ID='LED1' Name='LED1' Value='LED1' />

LED Node 2
<INPUT Type='radio' ID='LED2' Name='LED2' Value='LED2' /> <br/><br/>

<INPUT Type='text' ID='ADC' Value='' /> <SPAN ID='unit' >ADC-Value</SPAN> <br/>

<BUTTON Type='button' onclick='javascript:SetSensorScale(RESISTANCE)'>Ohm</BUTTON>
<BUTTON Type='button' onclick='javascript:SetSensorScale(RAWVALUE)'>Adc raw</BUTTON> <br/><br/>

<BUTTON Type='button' onclick='javascript:GotoUrl("Limit")'>Set-Limit-Page</BUTTON> <br/><br/>

</FORM>
```

Wichtige Funktionen der Javascript-Library JSBus

`function InitParts()`
Gibt ein leeres Array von Parts zurück (Aufruf: `var parts = Initparts();`).

```
function SetLimit(parts, sender, receiver, channel, mode, limit, numvalue)
function SetScale(parts, sender, receiver, channel, mode, quantity, unit, scale)
function SetValue(parts, sender, receiver, channel, mode, setvalue)
```
Diese Funktionen hängen an das Array `parts` jeweils ein weiteres Part an, das einen Grenzwert, die Messgröße/Einheit/Skala oder einen Soll-Wert auf einem Sensor/Aktor setzt, und geben das erweiterte Array zurück.

`function SendParts(parts, overrideQueue)`
Encodiert und versendet alle Parts in einer oder mehreren Messages, siehe Text.

`function PartText(part)`
Gibt eine Textdarstellung eines Parts zurück, zum Beispiel für Debugging-Zwecke.

`function RadioButtonSetvalue(id, setvalue)`
Setzt oder resettet einen Radiobutton (`setvalue` = 1 oder 0).

```
function TextboxSetvalue(id, setvalue)
function TextSetvalue(id, setvalue)
```
Beschreibt eine Textbox oder ein Text-Element mit Text.

`function GotoUrl(url)`
Veranlasst den Host, eine neue HTML-Seite zu laden (`url` = Dateiname ohne „.htm“).

`function SetScheduler(status, schedulednode1, … , schedulednode8)`
Schaltet auf dem Host den Scheduler an/aus (`status` = `SCHEDULER_ON` / `SCHEDULER_OFF`) bzw. teilt dem Scheduler eine neue Liste von Knoten mit, die regelmäßig zum Senden aufgefordert werden sollen (eine 0 kennzeichnet das Ende der Liste).

leitet von Tags wie „DIV“ oder „INPUT“. Für uns besonders wichtig sind die INPUT- und BUTTON-Elemente. Bei den INPUT-Elementen erkennt man den Typ (Radio[button] oder Text[box]) und die ID, mit der das Eingabe-Element eindeutig gekennzeichnet ist, um es später von Javascript aus beschreiben oder auslesen zu können. Text, den wir zur Laufzeit ändern wollen, steckt man am besten in <DIV>- oder <SPAN>-Tags, wobei Ersteres gleich für einen eigenen Absatz sorgt.

Die zusätzlichen, unterschiedlichen *Name*-Attribute im Radiobutton-Element (LED1 und LED2) sorgen dafür, dass die Buttons unabhängig voneinander angesteuert werden können. Bei einem identischen *Name*-Attribut darf nämlich immer nur einer der Knöpfe gleichzeitig betätigt sein, eben wie bei einem richtigen Radio älteren Jahrgangs.

Das *onclick*-Attribut der Buttons verknüpft den Klick mit der aufzurufenden Javascript-Funktion. Klicks auf die ersten beiden Buttons schalten den Scheduler über die Funktion `SetScheduler` (in JSBus.txt) an oder aus. Die weiter unten definierten Buttons rufen die eben beschriebene, anwendungsspezifische Funktion `SetSensorScale` mit dem entsprechenden Parameter (`RESISTANCE` oder `RAWVALUE`) auf. Der letzte Button sorgt dafür, dass der Browser das Formular „Limit.htm“ lädt. Hierzu wird die Funktion `GotoUrl` in der JSBus-Library genutzt. Als Parameter wird der Dateiname (ohne „.htm“) übergeben. Man beachte, dass man den Namen in diesem Fall unbedingt in doppelte Anführungszeichen einschließen muss, da einfache Anführungszeichen schon für den Wert des *onclick*-Attributs verwendet werden. Die Dateien müssen sich im Ordner „UIBus“ auf dem Desktop befinden. Spätere Versionen des ElektorBusBrowsers sollen hierfür noch Einstellmöglichkeiten mitbringen.

Ausblick

Wer schon ein paar HTML/Javascript-Vorkenntnisse hat, dürfte mit diesen Informationen und dem Kasten, der die wichtigsten Funktionen der JSBus-Bibliothek beschreibt, schon zu ansehnlichen eigenen Applikationen kommen. Anfänger sollten sich die „HTML/Javascript-Basics“ (Download unter [2]) zu Gemüte führen und dann einfach mal ausprobieren, was Veränderungen des HTML-Codes bewirken. Zur Eingabe genügt ein einfacher Texteditor, mit einem Doppelklick auf die .htm-Datei öffnet sich die Datei im Lieblingsbrowser. Damit sieht man immerhin schon einmal, wie die Nutzeroberfläche später aussehen wird. Und da wir das System für die weiteren Teile der Bus-Serie

benutzen werden, wird schnell weiterer Javascript-Code hinzukommen, den man in eigenen Anwendungen verwenden kann.

Die Beschäftigung mit dem Gespann Javascript/HTML ist in jedem Fall sehr nützlich, schon allein weil wir in einem der nächsten Teile der Serie dasselbe Konzept verwenden werden, um unsere Anwendung von einem Android-Smartphone aus zu steuern (**Titelbild**). Als Brücke zum Bus kommt eine clevere Platine auf Basis des Vinculum-II-Chips von FTDI zum Einsatz, die wir im Januar-Heft vorstellen wollen. Diese Platine eignet sich aber nicht nur für den ElektorBus, sondern auch für andere Anwendungen. Javascript/HTML bietet sich dann ebenfalls an, um einfach zu einem plattformunabhängigen User-Interface zu kommen.

(110517)

Entwickeln Sie mit!
Hinweise und Ideen sind willkommen unter redaktion@elektor.de!

Weblinks

[1] www.elektor.de/110258
[2] www.elektor.de/110517
[3] www.elektor.de/110428
[4] www.elektor.de/110382
[5] www.w3schools.com
[6] http://de.selfhtml.org/

Ein neues Bit

Während der Arbeit an diesem Artikel bekam ich wieder Hinweise von Elektor-Lesern, die teilweise auch eigene Entwicklungen beisteuern. So wie Jan Dalheimer aus Schweden, der mit 15 Jahren wohl einer der jüngsten Bus-Fans sein dürfte. Zum Redaktionsschluss war Jan gerade dabei, eine ElektorBus-Bibliothek für AVR-Controller zu implementieren. Seine erste Variante behandelte zuerst einmal nur das *ElektorMessageProtocol* und den *HybridMode*, nicht aber das in den beiden letzten Teilen vorgestellte *ApplicationProtocol*. Dabei fiel Jan eine Unstimmigkeit des bisher beschriebenen Acknowledge-Mechanismus auf. Zwar ist es möglich, mit Hilfe des *ElektorMessageProtocols* eine Nachricht so zu kennzeichnen, dass eine Bestätigungsnachricht vom Sender erwartet wird (über Bit0 des Mode-Bytes, siehe [1]). Jedoch konnte eine Bestätigungsnachricht (die sinnvollerweise die empfangenen Datenbytes zum Sender zurücksendet) auf dieser Ebene nicht von der Originalnachricht unterschieden werden, denn das entsprechende Flag hatten wir erst mit dem *ApplicationProtocol* eingeführt. Damit erstreckt sich eine Funktion des Bus-Systems aber über zwei Protokoll-Schichten, was gerade beim sauberen Implementieren einer Bibliothek ein Nachteil ist.

Jan machte daher den Vorschlag, Bit1 des Modebytes zur Unterscheidung einer *AcknowledgeMessage* von der Originalnachricht zu verwenden. Eine gute Idee, die wir gerne aufgreifen – als Möglichkeit, eine *AcknowledgeMessage* auf Message-Ebene zu kennzeichnen. Da Bit1 des Modebytes aber schon mit einer anderen Funktion belegt war, müssen wir die Funktionen nun neu ordnen:

Bit	1	0
7	keine ID-Bytes, Daten ab Byte 2	ID-Bytes ab Byte 2
6	Byte 2 und 3 sind ID-Bytes	Byte 2 bis 5 sind ID-Bytes
5	keine CRC	Byte E und F sind 16-bit-CRC
4	letztes ID-Byte ist Fragmentnummer	Alle ID-Bytes für Adressierung
3	Nächstes Fragment folgt direkt	kein Fragment folgt direkt
2	höchste sechs Adressbits für Bus-Segment	keine Segment-Adresse
1	AcknowledgeMessage	Original-Nachricht
0	AcknowledgeMessage erwartet	nicht erwartet

Bit1 und Bit0 sind in Bild 5 rot umrandet. In der Javascript-Library ist der neue Vorschlag schon umgesetzt: Eine Bestätigungsnachricht wird automatisch mit einem Modebyte=2 herausgeschickt, wenn eine Originalnachricht mit einem Modebyte=1 empfangen wurde.

Hiervon zu unterscheiden ist der Acknowledge-Mechanismus auf der Ebene des *ApplicationProtocols*; ein Beispiel bietet die Alarm-Meldung unseres Fotosensors (siehe [3]). Da dieser Knoten regelmäßig abgefragt wird, ist das Modebyte der von ihm gesendeten Messages normalerweise 0 (keine Bestätigung nötig, da keine Kollisionen auftreten können). Die Bestätigung muss der Anwendungsentwickler dann selbst implementieren. Zur Unterscheidung von der Originalnachricht wird das in [4] eingeführte Ack-Flag innerhalb der Nutzdaten verwendet. Ein wichtiger *Part* innerhalb einer *Message* (zum Beispiel eine Alarm-Meldung) wird dazu einfach mit einem Ackflag=true versehen und zu einer neuen Message encodiert, die anschließend zurückgesendet wird. Hierzu hält die Javascript-Library die Funktion `AddAckPart(…)` bereit, ein Beispiel zur Verwendung findet man in der Datei „Limit.htm“.

Gradwanderung

Ein Messgerät für Temperaturänderungen

Von Dr. Dietmar Schröder (D)

Manchmal interessieren nicht so sehr die Absolutwerte der Temperatur, sondern kleinste Temperaturänderungen. Die hier vorgestellte Schaltung spürt sie nicht nur auf, sondern macht sie auch sichtbar und hörbar. Dabei kommt sie mit wenigen aktiven Bauteilen aus und zeigt eine gemessene Temperatur mit einer Auflösung von einem zehntausendstel Grad an.

Temperaturen messen ist relativ einfach, sie präzise zu messen jedoch eine Wissenschaft für sich. Um Temperaturen genauer als 1/10 Grad zu bestimmen, ist ein erheblicher Aufwand nötig. Die hier vorgestellte Schaltung misst Temperaturen mit einer Auflösung von einem zehntausendstel Grad und besteht aus nur vier aktiven Bauteilen sowie einem optionalen Display. Der Zweck dieser Schaltung besteht nämlich darin, feinste Temperaturänderungen aufzuspüren. Dabei wird bei dieser Schaltung weniger auf die absolute Genauigkeit Wert gelegt, sondern vielmehr auf die unglaubliche Empfindlichkeit durch eine bemerkenswert hohe Auflösung.

Eigenschaften und Möglichkeiten

Das Messgerät misst die Temperatur ohne aufwendigen Abgleich nur auf rund zwei Grad genau, es zeigt aber zusätzlich deren Steigung, also den Grad der Änderung an. Neben den Zahlenwerten auf einem Display erfolgt die Anzeige auch analog durch ein Zeigerinstrument und akustisch über eine sich verändernde Tonhöhe. Die Messwerte können auch über eine serielle Schnittstelle zu einem PC übertragen werden.

Mit solch einem Messgerät kann man hochinteressante Untersuchungen machen. Die Empfindlichkeit ist so hoch, dass sich die Anzeige schon deutlich ändert, wenn man den Temperaturfühler hebt oder senkt - in einem Raum ist es oben etwas wärmer als unten.

Positioniert man den Fühler vor der Ausblasöffnung eines PC-Gehäuses, so kann man die Auslastung von CPU und Grafikprozessor an der Temperaturänderung erkennen.

An einem Wasserrohr im Haus erkennt man an der Temperaturänderung, ob jemand Wasser verbraucht.

Die akustische Ausgabe ermöglicht eine Überwachung im Hintergrund. Klemmt man den Fühler an ein empfindliches Bauteil in einer Schaltung, so kann man es hören, wenn dort plötzlich mehr Leistung umgesetzt wird. Man muss den Fühler gut isolieren, um die Anzeige ruhig zu halten, zum Beispiel, indem man ihn tief in einem Stück Isolierschaum (Styropor) versenkt. Wenn man die Zeitkonstante der Steigungsmessung entsprechend wählt, kann man trotz der Isolierung die langsame Änderung der Raumtemperatur problemlos erkennen.

Der Autor hat die Schaltung ursprünglich entwickelt, um an einem PKW das Verhalten des Thermostatventils zwischen innerem und äußerem Kühlkreislauf zu analysieren.

Aufbau

Der Temperaturmesser besteht im Wesentlichen aus einem NTC als Temperaturfühler, einer Spannungsreferenz, einem A/D-Wandler und einem Mikrocontroller mit LC-Display.

Leserprojekte sind Beiträge von Elektor-Lesern für experimentelle Zwecke oder zur Anregung für andere Leser.
Die in dieser Rubrik vorgestellten Schaltungen wurden vom Elektor-Labor nicht auf Reproduzierbarkeit und Funktion getestet.

Eigenschaften

Arbeitsbereich: 0..40 °C
Absolute Genauigkeit: ca. 2 °C ohne Kalibrierung
Auflösung: theoretisch 0,04 Milligrad, praktisch ca. 0,3 Milligrad (Anzeige: 0,1 Milligrad)
Messbereich: Grad je 2 Sekunden...Grad je 100 Minuten
Versorgungsspannung: 6 V (4 x Mignon)

Um Störeinflüsse zu minimieren, muss der NTC möglichst unmittelbar mit dem Analog/Digital-Wandler verbunden sein. Damit man den Fühler trotzdem an einem längeren Kabel betreiben kann, wurde die Schaltung auf zwei Platinen aufgeteilt.

Der Fühler ragt aus einer sehr schmalen Fühlerplatine heraus, auf der sich auch die Referenzspannungsquelle und der A/D-Wandler befinden. Der Fühler ist dabei nur etwa 10 mm vom Wandler entfernt. Als Gehäuse dient ein Schrumpfschlauch, und das Ganze bildet dann die Messspitze (**Bild 1**) des Messgeräts.

Die Messdaten werden digital über einen SPI-Bus mit einem 4-adrigen Anschlusskabel zur Prozessorplatine (**Bild 2**) mit einem ATmega88-Mikrocontroller übertragen. An diese Platine können ein Display, ein Piezo-Schallwandler, ein analoges Zeigerinstrument und eine serielle Schnittstelle (beziehungsweise ein USB-Seriell-Schnittstellenkabel) mit TTL-Pegel angeschlossen werden. Mit drei Potentiometern lassen sich die Empfindlichkeit, die Zeitkonstante und die Lautstärke des Piezo-Schallwandlers einstellen.

Schaltung

Die Schaltung der Sensorplatine ist in **Bild 3** dargestellt. Die 5-V-Versorgungsspannung wird über das vierpolige Verbindungskabel von der Prozessorplatine geliefert und durch ein RC-Glied mit R100 und C100 gefiltert. Die Referenzspannungsdiode TL431 erzeugt daraus eine rauscharme Spannung von etwa 2,5 V, die durch R102 und C103 nochmals gefiltert wird und als Referenzspannung für den Analog-Digital-Wandler MCP3551 dient. Die genaue Höhe der Referenzspannung ist dabei nicht wichtig, denn der Wandler bewertet nur das Verhältnis des Widerstandes R103 zum NTC. R103 ist dabei so ausgelegt, dass der Verlauf der Kennlinie für die resultierenden ADC-Werte im Bereich von 0...40 °C nahezu linear verläuft (**Bild 4**). Der Kondensator C104 am Eingang des Wandlers dient als Antialiasing-Filter.

Der MCP3551 ist ein Delta-Sigma-Wandler mit 22 bit Auflösung. Im Bereich um 20 °C bedeutet das in dieser Schaltung etwa 24.000 Schritte pro Grad. Das entspricht einer (theoretischen) Auflösung von 0,04 Milligrad. Praktisch ist sie durch das Rauschen der Bauteile etwa acht Mal geringer, beträgt also etwa 19 bit.
Der MCP3551 misst ca. 20.000 Mal pro Sekunde die Spannung am NTC und filtert die Werte digital. Dabei ist in dem Chip neben einem Tiefpass auch ein hochwertiges Kerbfilter integriert, mit dem 50...60-Hz-Brummen wirkungsvoll unterdrückt wird. An der SPI-Schnittstelle steht dann etwa 14 Mal pro Sekunde ein neuer Messwert bereit, der über das 4-polige Verbindungskabel zur Prozessorplatine gelangt.

In der Schaltung der Prozessorplatine in **Bild 5** erfolgt der Anschluss der 4-poligen Leitung von der Sensorplatine am Steckverbin-

Bild 1. Die Sensorplatine in der Messspitze stellt einen Temperaturfühler mit SPI-Ausgang dar.

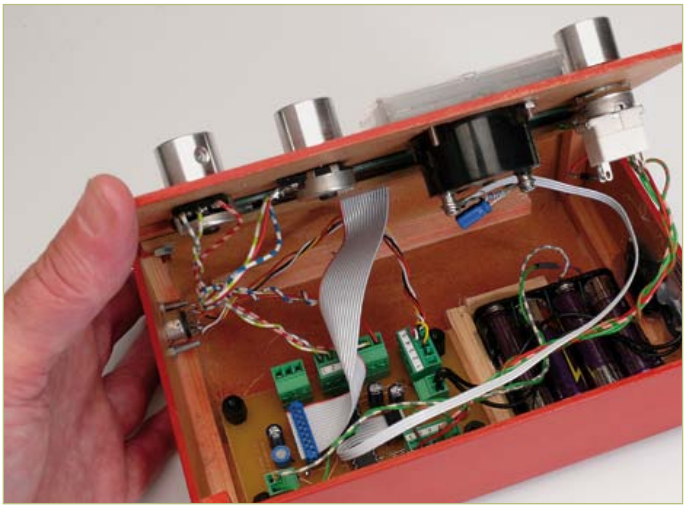


Bild 2. Die Prozessorplatine verarbeitet die SPI-Daten und steuert neben einem LCD auch ein Zeigerinstrument an.

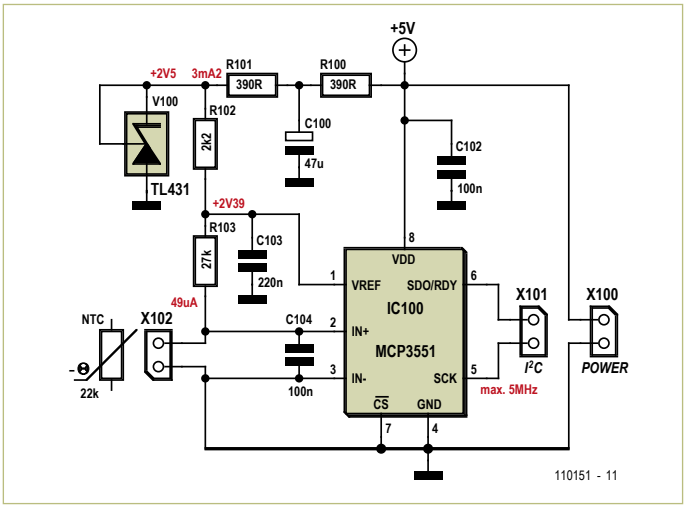


Bild 3. Schaltplan der Sensorplatine.

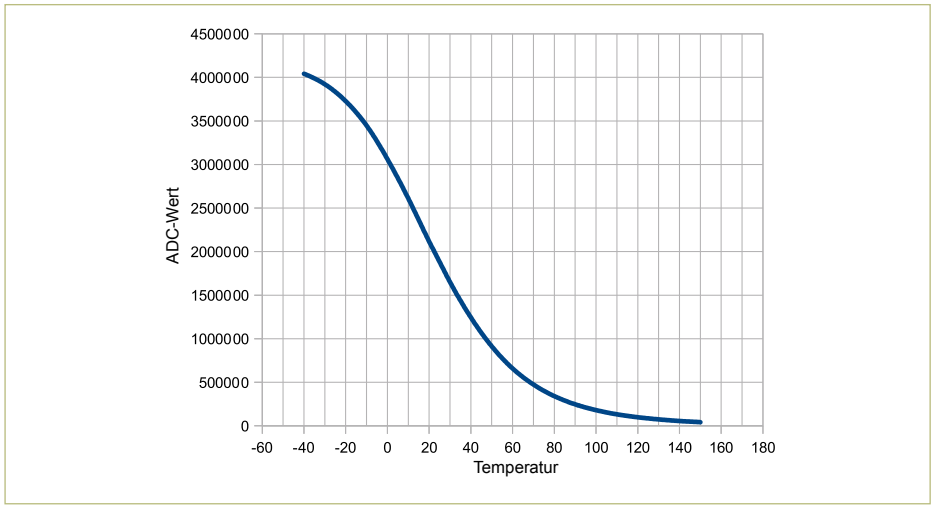


Bild 4. Kennlinie der Temperaturmessung.

der X202. Die Anschlüsse X201 und X205 sind für die Potentiometer P1 (10 k) und P2 (10 k) vorgesehen, mit denen sich die Zeitkonstanten der digitalen Messwertverarbeitung in der Controller-Software einstellen lassen. An den Prozessor sind neben dem Sensor ein LC-Display und zwei Potentiometer (P1 und P2) angeschlossen, über die sich Zeitkonstanten einstellen lassen. Ein drittes Potentiometer stellt die Lautstärke des Piezo-Schallwandlers (ohne Elektronik, ca. 8..20 nF) ein. Der Poti-Anschluss X206 ist 5-polig, da ein 10-k-Poti mit Schalter verwendet wird, das gleichzeitig als Ein-/Ausschalter dient. Ein viertes Poti (P500) ist für die Kontrasteinstellung des an X207 angeschlossenen LC-Displays zuständig. Der Autor hat dafür ein LCD-Modul des

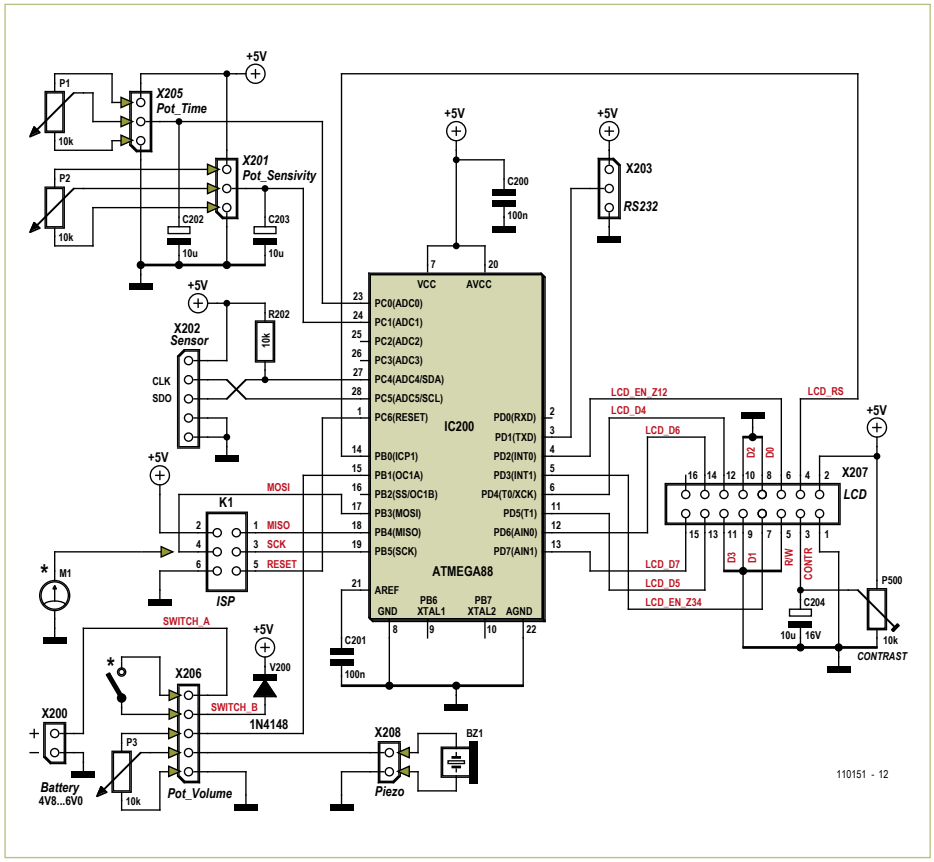


Bild 5. Schaltplan der Prozessorplatine.

Typs Wintek WD-C2704M verwendet, ein 4x27-Zeichen-Display mit zwei HD44780-kompatiblen Controllern, das sich im 4-bit- oder 8-bit-Modus ansteuern lässt.

Da der ATmega88 hier den internen 1-MHz-Oszillator verwendet, ist kein Quarz vorhanden. Die Programmierung erfolgt über den ISP-Anschluss X204. Der MOSI-Pin dieses Steckverbinders dient auch als Anschluss für ein Zeigerinstrument (Messbereich 0...5 V). Versorgt wird die Prozessorplatine aus vier AA-Zellen (Mignon-Batterien oder Akkus). Die Diode am Eingang dient als Verpolschutz und reduziert die Batteriespannung um etwa 0,6 V, damit am Prozessor und A/D-Wandler die maximal zulässige Betriebsspannung von 5,5 V nicht überschritten wird.

Die Platinenlayouts (**Bild 6** und **Bild 7**) zum Aufbau der beiden Schaltungen stehen im PDF-Format und als Protel-Datei unter [1] zum kostenlosen Download zur Verfügung.

Software

Auch die Controller-Firmware mit Source- und Hexcode kann von der Elektor-Webseite zu diesem Leserprojekt [1] kostenlos heruntergeladen werden.

Die Software liest zyklisch die von der Sensorplatine gelieferten Messwerte und rechnet sie in Grad Celsius um. Dieser Wert wird als ASCII-Text über die serielle Schnittstelle ausgegeben (z.B. „23,5341°C“). Die Kennlinie des Temperaturfühlers ist als Tabelle in 5-Grad-Schritten hinterlegt, zwischen denen linear interpoliert wird. Damit kann die Temperatur im Bereich von 0-40 °C auf rund 2 Grad genau bestimmt werden. Die Ungenauigkeit ergibt sich durch die Parameterstreuung unterschiedlicher NTCs. Mit einem kalibrierten NTC kann man genauer als 0,5 Grad messen.

Die Temperaturwerte werden durch zwei in der Software realisierte und hintereinander geschaltete Tiefpässe (TP1 und TP2) geleitet. TP1 filtert kurzzeitige Schwankungen heraus. Die Temperaturänderung ergibt sich aus der Differenz zwischen dem Wert vor und hinter TP2.

Die mit den Potis an P1 und P2 eingestellten Spannungswerte werden mit dem A/D-Wandler des ATmega gemessen und daraus Zeitkonstanten τ1 und τ2 für TP1 und TP2 bestimmt. Die Tiefpässe funktionieren wie ein einfaches RC-Glied, sie erreichen also bei einem Spannungssprung nach der Zeit 5τ 99 % ihres Endwertes. Dabei wird der eingestellte Wert quadriert, so dass sich Zeitkonstanten sowohl im Sekundenbereich als auch im Minutenbereich einstellen lassen. Da es keinen Sinn macht, die Zeitkonstante für TP2 kleiner als die von TP1 zu machen, wird die Filterzeit von TP1 zu TP2

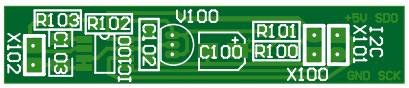


Bild 6. Bestückungsplan der Sensorplatine.

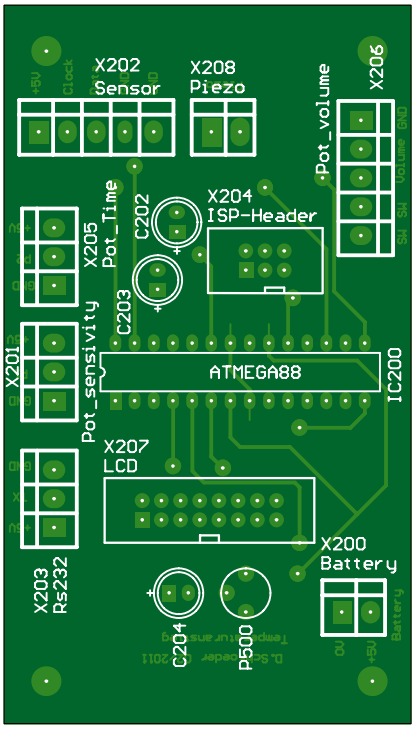


Bild 7. Die Bestückung der Prozessor-Platine.

Bild 8. Die auf dem 4-zeiligen Display gezeigten Werte.

hinzuaddiert. Der einstellbare Bereich geht damit für τ1 von 0 s bis 19 min und für τ2 von 2 s bis 100 min. Im Allgemeinen sollte der Unterschied zwischen TP1 und TP2 mindestens Faktor 2 sein.
Eine kleine Hysterese verhindert ein Schwanken der Einstellwerte.
Die Temperaturänderung wird über einen PWM-Ausgang auf dem Zeigerinstrument visualisiert. Dazu wird zusätzlich die Batteriespannung ermittelt und das PWM-Tastverhältnis so eingestellt, dass sich bei Temperaturänderung Null eine Spannung von 2,5 V ergibt. Bei einem 5-V-Zeigerinstrument steht der Zeiger dann genau in der Mitte.
Der PWM-Ausgang ist so ausgelegt, dass der Zeiger einen Bereich von ±50 Milligrad pro Sekunde anzeigt.

Zwei Timer erzeugen einen Ton, dessen Tonhöhe sich je nach Gradient der Temperatur ändert. Dabei wird bei fallenden Temperaturen der Ton mit dem zweiten Timer periodisch unterbrochen, so dass sich statt eines Dauertones ein Piep-Piep-Piep ergibt. Solange sich die Temperatur nur innerhalb der Rauschgrenzen ändern, wird der Ton abgeschaltet.

Auf dem angeschlossenen Display wird eine Tabelle angezeigt, welche die Temperatur, die eingestellten Zeitkonstanten und die ermittelte Temperatursteigung enthält (**Bild 8**).

In der zweiten Zeile wird die aktuelle Temperatur mit vier Nachkommastellen angezeigt, wobei die letzte Stelle nur noch bedingt aussagekräftig ist, da sie in der Größenordnung des Rauschens der Messschaltung liegt. Bei guter Isolierung des Messfühlers schwanken die Werte der letzten Stelle um etwa 3 bis 4.

Die dritte und vierte Zeile geben die Temperatur nach Filterung durch die Tiefpässe an. Die mit den Potentiometern eingestellten Zeitkonstanten werden dabei in der 2. Spalte angezeigt. Die letzte Spalte enthält nun die eigentlich interessanten Werte: Die Temperaturänderungen. Der Wert der dritten Zeile ist die Differenz zwischen aktuellem Messwert und Tiefpass 1, der Wert der vierten Zeile die Differenz zwischen Tiefpass 1 und Tiefpass 2.

Inbetriebnahme

Für den Anfang sollten P1 und P2 ganz nach links gedreht werden, was 0 s für TP1 und 2 s für TP2 ergibt. Anschließend empfiehlt es sich, etwas zu warten, während der Messfühler sich durch den Messstrom erwärmt. Das ist nicht zu vermeiden, denn im NTC werden etwa 0,2 Milliwatt Leistung verheizt, was zu einem minimalen Temperaturanstieg führt. Dann kann man eine Hand ein paar Sekunden in etwa 10 cm über den Fühler halten und beobachten, wie der Zeiger des Drehspulinstruments auf Grund der Wärmestrahlung deutlich nach rechts ausschlägt, um anschließend nach links auszuschlagen, wenn sich der Fühler wieder abkühlt.

Der Autor verwendet für die meisten Messungen einen Wert von 5-10 s in der 3. LCD-Zeile (TP1) und von 20-120 s in der 4. LCD-Zeile (TP2).

(110151)

[1] www.elektor.de/110151

Der Autor

Dr.-Ing. Dietmar Schröder arbeitet als Software- und Hardwareentwickler bei aixcon Powersystems GmbH und betreibt eine eigene Homepage mit (auch ungewöhnlichen) Selbstbauprojekten unter www.zabex.de.

Fledermaus-Sonar

Hoch empfindlich, schnell gebaut

Von Jan van Eck (NL) j.vaneck@fontys.nl

Europäische Tierschützer hatten das Jahr 2011 zum Jahr der Fledermaus erklärt. Damit wollten sie Aufmerksamkeit auf die manchmal mystisch umwobenen, vielerorts unbekannten fliegenden Säugetiere lenken. Inzwischen ist die kalte Jahreszeit ins Land gezogen, die Fledermäuse halten Winterschlaf. Das ist die richtige Zeit, um ein Fledermaus-Sonar zu bauen. Das nächste Frühjahr kommt bestimmt – und der eigenen Fledermaus-Forschung steht dann nichts mehr im Wege.

Fledermäuse senden akustische Signale aus, die das menschliche Gehör nicht wahrnehmen kann. Die Frequenzen liegen bei vielen heimischen Arten im Bereich des Ultraschalls um 40 kHz. Wenn die Schallsignale auf ein Hindernis treffen, werden sie reflektiert. An den reflektierten Signalen erkennt die Fledermaus die Entfernung und Größe des Objekts. Um Ultraschallsignale in den hörbaren Bereich zu verschieben, müssen sie zuerst von einem Mikrofon empfangen werden. Dann teilt eine elektronische Schaltung die Signalfrequenz so weit herab, dass sie in den vom Menschen hörbaren Bereich fällt. Dieses Signal wird von einem Lautsprecher wiedergegeben.

Schaltung

Als Mikrofon dient ein Ultraschallempfänger des Typs 400SR160 von Prowave. Wir haben die preiswerte Ausführung im Kunststoffgehäuse gewählt, sie kann mit Alufolie oder leitfähigem Klebeband gegen Störsignale abgeschirmt werden. Das von diesem sogenannten „Transducer“ kommende Signal wird von einem LM386 (IC1) etwa 200-mal verstärkt (**Bild 1**). Der LM386 ist ein kleiner, preiswerter Endverstärker, er leistet aber auch als einfacher NF-Verstärker gute Dienste. Abgesehen von zwei Entkoppelkondensatoren (C2 und C4) kommt dieses IC normalerweise ohne externe Bauelemente aus. Wenn die interne Gegenkopplung durch einen Kondensator an den Pins 1 und 8 überbrückt wird, steigt die Verstärkung von mindestens 26 dB auf 46 dB an. Der Kondensator ist üblicherweise ein Elko mit der Kapazität 10 µF, das Audio-Frequenzband wird dann ungefähr gleichmäßig verstärkt. Hier ist eine lineare Verstärkung nicht notwendig, so dass 220 nF für C1 genügen. Da bei diesem Wert die Signalan-

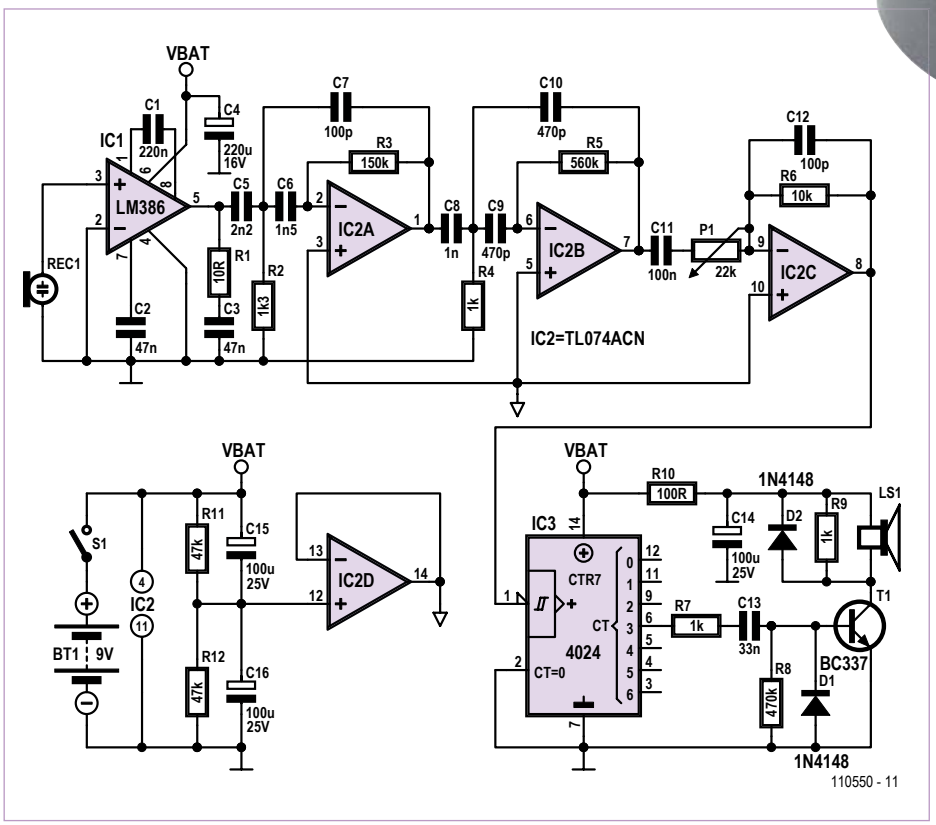
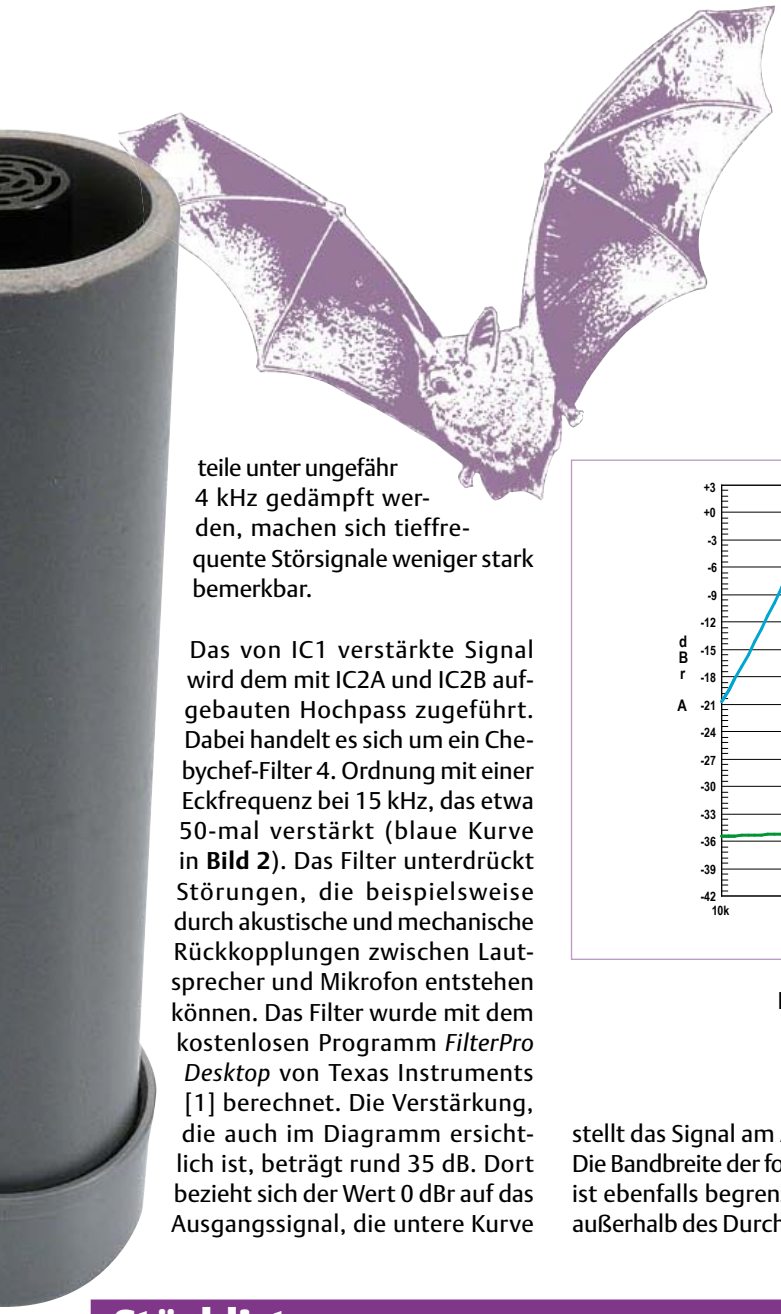


Bild 1. Die Schaltung des Fledermaus-Sonars ist ohne Geheimnisse.



teile unter ungefähr 4 kHz gedämpft werden, machen sich tieffrequente Störsignale weniger stark bemerkbar.

Das von IC1 verstärkte Signal wird dem mit IC2A und IC2B aufgebauten Hochpass zugeführt. Dabei handelt es sich um ein Chebychef-Filter 4. Ordnung mit einer Eckfrequenz bei 15 kHz, das etwa 50-mal verstärkt (blaue Kurve in **Bild 2**). Das Filter unterdrückt Störungen, die beispielsweise durch akustische und mechanische Rückkopplungen zwischen Lautsprecher und Mikrofon entstehen können. Das Filter wurde mit dem kostenlosen Programm *FilterPro Desktop* von Texas Instruments [1] berechnet. Die Verstärkung, die auch im Diagramm ersichtlich ist, beträgt rund 35 dB. Dort bezieht sich der Wert 0 dBr auf das Ausgangssignal, die untere Kurve stellt das Signal am Ausgang von IC1 dar. Die Bandbreite der folgenden Stufe mit IC2C ist ebenfalls begrenzt, so dass Störsignale außerhalb des Durchlassbereichs zusätzlich gedämpft werden. Mit dem für C12 angegebenen Wert beträgt die Bandbreite etwa 160 kHz, falls notwendig kann sie noch weiter herabgesetzt werden.

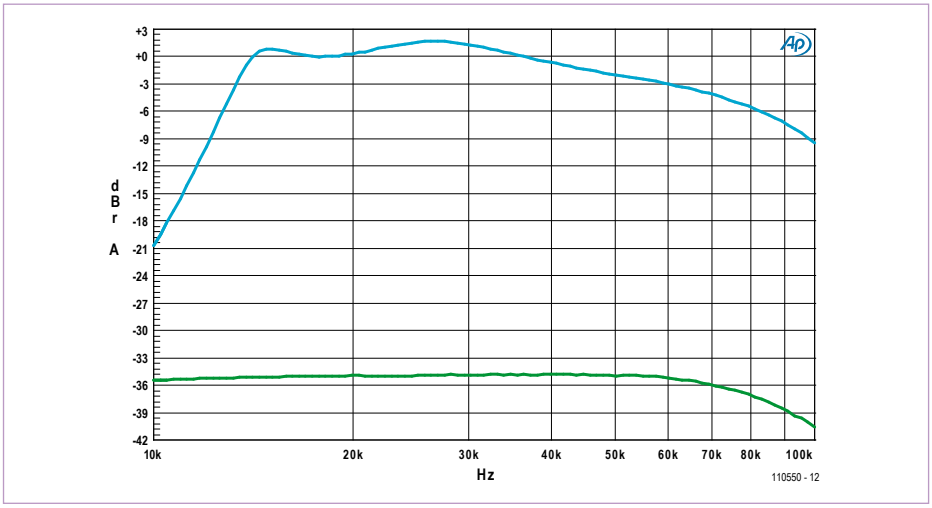


Bild 2. Frequenzgang am Ausgang von IC2B (oben) und am Ausgang von IC1 (unten).

Stückliste

Widerstände (0,25 W, 5 %):
R1 = 10 Ω
R2 = 1k3
R3 = 150 k
R4,R7,R9 = 1 k
R5 = 560 k
R6 = 10 k
R8 = 470 k
R10 = 100 Ω
R11,R12 = 47 k
P1 = 22 k Trimmpoti stehend

Kondensatoren:
C1 = 220 n MKT, Raster 5 mm
C2,C3 = 47 n MKT, Raster 5 mm
C4 = 220 µ/16 V stehend, Raster 2,5 mm
C5 = 2n2 MKT, Raster 5 mm
C6 = 1n5 MKT, Raster 5 mm
C7,C12 = 100 p keramisch, Raster 5 mm
C8 = 1 n MKT, Raster 5 mm
C9,C10 = 470 p keramisch, Raster 5 mm
C11 = 100 n MKT, Raster 5 mm
C13 = 33 n MKT, Raster 5 mm
C14,C15,C16 = 100 µ/25 V stehend, Raster 2,5 mm

Halbleiter:
D1,D2 = 1N4148
T1 = BC337-40
IC1 = LM386N-3
IC2 = TL074CN
IC3 = 4024

Außerdem:
REC1 = Ultraschallempfänger 40 kHz, ∅ 16 mm (z.B. Prowave 400SR16P)
LS1 = Kleinlautsprecher 8 Ω/0,3 W, ∅ 20 mm (z.B. Kingstate KDMG20008)
S1 = Schiebeschalter 1-polig
BT1 = Batterie, 9 V, mit Anschlussclip
3 x 2-polige Stiftleiste, Raster 2,54 mm, für LS1, S1 und BT1
Platine 110550-1 (siehe www.elektor.de)

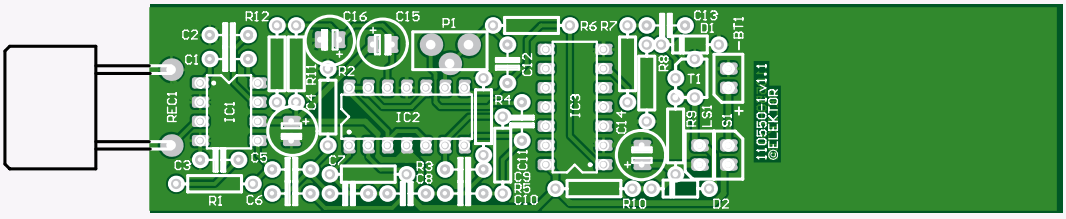
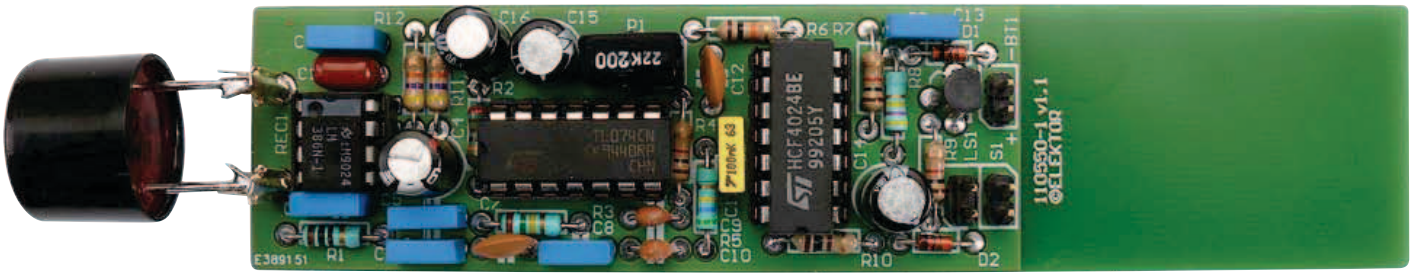


Bild 3. Die Platine ist schmal und schlank, so dass sie in ein Stück Kunststoffrohr passt.



Der vierte Opamp des TL074 (IC2) wird genutzt, um zusammen mit einem Spannungsteiler (R11, R12) das Massepotential der übrigen Opamps auf die halbe Betriebsspannung festzulegen.

IC3 ist der Frequenzteiler, der das Fledermaus-Signal in den für das menschliche Ohr hörbaren Bereich umsetzt. Am Ausgang CT3 (Pin 6) des siebenstufigen Binärzählers 4024 erscheint das durch 16 frequenzgeteilte Eingangssignal. Es liegt daher im Bereich von etwa 2...3 kHz und wird von T1 verstärkt, bevor es zum Lautsprecher LS1 gelangt. Die Diode D1 schützt die Basis von T1 vor negativen Spannungen, R7 begrenzt den Basisstrom. Weil sich der Lautsprecher induktiv verhält, liegt Diode D2 parallel, um T1 vor Spannungsspitzen zu schützen. Auch Piezo-Schallwandler sind als Lautsprecher verwendbar. Da diese Schallwandler kapazitiven Charakter haben, ist R9 hinzugefügt, um die Wandlerkapazität zu entladen. Unerwünschten Kopplungen vom Lautsprecherausgang über die Betriebsspannung zurück zum Eingang wirken C14 und R10 entgegen. Der Strombedarf des Fledermaus-Sonars beträgt im Ruhezustand etwa 14 mA und steigt beim Empfang von Ultraschallsignalen auf maximal etwa 90 mA. Die Mindest-Battteriespannung ist etwa 4,5 V.

Miniplatine

Das Platinenlayout für das Fledermaus-Sonar ist in **Bild 3** wiedergegeben. Die untere Platinenhälfte dient als Montagefläche für die 9-V-Batterie. Auf der Platinenoberseite schirmt eine Massefläche die empfindliche Eingangsschaltung ab. Die Platine ist so bemessen, dass sie in ein Kunststoffrohr mit einem Durchmesser von 33...35 mm eingeschoben werden kann. In Baumärkten sind für solche Rohre auch passende Abschlussstopfen erhältlich. In einen Abschlussstopfen können, wie **Bild 4** zeigt, der Lautsprecher und der Schalter eingebaut werden. Drei Klebefüße halten Abstand zur Standfläche, zum Beispiel wenn das Fledermaus-Sonar senkrecht auf dem Gartentisch oder auf einer Mauer steht.

Mit Trimmpotentiometer P1 ist die Empfindlichkeit so einzustellen, dass der Lautsprecher ohne Ultraschall-Empfang gerade noch schweigt oder ein fast unhörbares Geräusch von sich gibt. Fremde Ultraschallquellen wie Leuchtstofflampen, TV-Geräte, Computer oder andere Geräte mit Schaltnetzteilen stören die Fledermaus-Signale, solche Geräte dürfen sich nicht in unmittelbarer Nähe befinden. Das Fledermaus-Sonar reagiert bereits auf den Ultraschall, der beim Aneinanderreiben von Daumen und Zeigefinger, beim leisen Knüllen einer Plastiktüte oder beim verhaltenen Schwenken eines Schlüsselbunds entsteht. Da Fledermäuse zu lautstarkem (Ultraschall-) Geschrei neigen, können sie über Distanzen von 30 Metern und mehr ausgemacht werden.

(110550)gd

Weblink

[1] www.ti.com/tool/filterpro

Bild 4. Der Schalter und der Lautsprecher sind im Rohrstopfen montiert.

Elektronische Stimme

Wollen Sie ihrem nächsten Projekt das Sprechen beibringen?

Von Christian Picard (Frankreich)

Wenn man abgelenkt ist, dann ist es schwierig, Meldungen auf einem Display rechtzeitig wahrzunehmen. Zum Beispiel wenn man sein ferngesteuertes Flugzeug, Schiff oder Auto beobachtet, das mit Telemetrie ausgestattet ist. Und in solchen Fällen kann es eine gute Idee sein, sich über wichtige Parameter wie Geschwindigkeit, Höhe oder Akkuspannung akustisch unterrichten zu lassen. Idealerweise sagt einem die Elektronik direkt, was los ist.

Sprachmodul

Die Sprachfähigkeiten entstammen dem Modul SOMO-14D [1] von 4D Systems. Es kann Dateien von maximal 2 GB großen microSD-Karten einlesen. Sein Audio-Ausgang treibt entweder einen Kopfhörer oder direkt einen kleinen Lautsprecher.
Die notwendige Steuerung wird mit einem ATmega8AU-Mikrocontroller realisiert, der mit 8 MHz getaktet ist. Seine Aufgabe besteht in der Ansteuerung des Sound-Moduls, sodass nach Analyse einer Zahl die entsprechende Sprachausgabe erfolgt. Das Sound-Modul ist über eine serielle 2-Wire-Verbindung angeschlossen und der Mikrocontroller selbst empfängt seine Befehle ebenfalls als Slave an einem I²C-Bus.
Die Schaltung kann daher als Sprachmodul über den I²C-Bus angesprochen werden und so beliebige Projekte um eine Sprachausgabe erweitern, indem beispielsweise Ganzzahlen oder Fließkommazahlen als String aus ASCII-Zeichen vorgelesen werden. Die Werte können auch negativ sein.
Man kann das SOMO-Modul natürlich in einer Anwendung auch direkt nutzen, indem über eine I²C-Verbindung gesteuert beliebige Sound-Dateien ausgegeben werden. Diese Strategie würde aber Ressourcen (Rechenleistung und Speicher) im Mikrocontroller der Anwendung verschlingen. Ein eigener Mikrocontroller für die Sprachausgabe macht die Sache schlanker und eleganter. Außerdem ist so eine Lösung wesentlich portabler und flexibler.

Firmware

Das Hauptprogramm des Mikrocontrollers besteht aus kaum mehr als einer Schleife, die einen eingegangenen String sofort analysiert, nachdem dieser als gültiges I²C-Data-Frame decodiert wurde. Innerhalb dieser Schleife löst jedes empfangene Zeichen einen Interrupt aus. Das Programm erkennt das Ende des Strings am C-typischen Null-Zeichen ($00), dem zwei Hex-Zeichen folgen. Diese geben die Nummer (den Namen) der Datei auf der microSD-Karte an, die eine Meldung wie die Bezeichnung der Messeinheit (z.B. Volt) beinhaltet.
Dann wird die Schleife verlassen.
Hierzu ein Beispiel: Angenommen, folgender Spannungswert soll ausgegeben werden:

–235,12 V. Der hierfür zu sendende ASCII-String baut sich wie folgt (hexadezimal) auf: $2d $32 $33 $35 $2c $31 $32 $00 $00 $69.
Sie haben sicher bemerkt, dass das Minuszeichen in ersten ASCII-Zeichen $2d steckt und das Komma durch $2c übertragen wird. Nun zu den letzten beiden Zeichen. Es muss auch die Einheit „Volt“ ausgegeben werden. Die beiden Zeichen $00 $69 (= 105 dezimal) ergeben die Nummer der Datei auf der SD-Karte, die „Volt“ enthält. Das gilt zumindest für die Dateien des Autors. In Ihrem System können Sie auch andere Bezeichnungen mit anderen Nummern verwenden. Man muss nur die passenden Sound-Dateien aufzeichnen.

Nochmals zurück zur String-Analyse: Zuerst wird überprüft, ob ein Zeichen zu einer Zahl gehört oder aber einer direkten Adresse beim SOMO-Modul entspricht. Wenn ein Null-Zeichen erkannt wird, sind die beiden nächsten Zeichen die Nummer einer Datei, deren Inhalt dann eingelesen wird. Wenn das erste Zeichen kein Null-Zeichen ist, wird der String als Zahl interpretiert. In diesem Fall überprüft das Programm, ob es sich um eine Ganzzahl oder um eine Fließkomma-Zahl handelt, ob sie positiv oder negativ ist sowie wie viele Stellen sich vor dem Komma befinden und wie viele danach (möglich sind 1, 2 oder 3 Stellen danach). Die Ergebnisse dieser Tests entscheiden dann darüber, welche Subroutinen angesprungen werden. Wenn empfangene Zeichen weder Dateinummern noch dezimale Werte sind, werden sie als Kommandos für das Sound-Modul verstanden, mit denen man zum Beispiel die Lautstärke einstellen kann ($FF $F0 bis $FF $F7).

Für Ausgaben in anderen Sprachen als Französisch müssen unter Umständen kleine Anpassungen gemacht werden, denn es gibt relevante Unterschiede zwischen französisch und deutsch bei der

Leserprojekte sind Beiträge von Elektor-Lesern für experimentelle Zwecke oder zur Anregung für andere Leser.
Die in dieser Rubrik vorgestellten Schaltungen wurden vom Elektor-Labor nicht auf Reproduzierbarkeit und Funktion getestet.

Ausgabe von Zahlen. Man bedenke nur, wie 100 (FR: cent, DE: ein-hundert) und 1000 (FR: mille, DE: ein-tausend) ausgesprochen werden.
Bei englischer Ausgabe sollte ein Dezimalpunkt statt eines Kommas genommen werden. Wenn man solche Feinheiten nicht berücksichtigt, klingt das komisch.

Hinweise

- Führende nichtsignifikante Nullen werden unterdrückt.
- Wenn ein String mit einer Null (ASCII $30) beginnt, wird die Zahl als Dezimalbruch mit 1 bis 3 Nachkommastellen interpretiert und vorgelesen.
- Wenn eine Zahl tatsächlich Null ist, muss der String $30 $00 sein.
- Bei einer Zahl ohne Einheit sollte dem abschließenden Nullzeichen ein $C0 folgen, damit keine Einheit gesprochen und die Sound-Ausgabe gestoppt wird.

Beispiele (nach deutscher Adaption)

12 sollte als „zwölf“ ausgegeben werden.
527 sollte die Ausgabe „fünf hundert sieben und zwanzig“ erzeugen.
13,689,222 ergibt gesprochen „dreizehn millionen sechs hundert neun und achtzig tausend zwei hundert zwei und zwanzig“.
0,155 ergibt „null komma eins fünf fünf“ und bei
−0,155 wäre „minus null komma eins fünf fünf“ korrekt.

Schaltung

In **Bild 1** ist das recht einfache Schaltbild zu sehen. Das Modul fällt mit 25 × 40 mm recht kompakt aus und kann daher sehr einfach in bestehende Anwendungen integriert werden. Bei der Bestückung der Platine gibt es ein paar kleinere Hürden. Das manuelle Einlöten des ATmega8AU-Controllers im TQFP32-SMD-Gehäuse klappt besser mit SMD-Lötpaste als mit Lötzinn. Die Methode des Autors: Das IC zunächst in Position bringen und mit einer Klammer fixieren. Dann etwas Lötpaste über eine Reihe von Pins verteilen. Bei Verwendung eines Lötkolbens mit feiner Spitze und der richtigen Menge Lötpaste treten keine Kurzschlüsse zwischen den Pins auf.
Zur Programmierung des Controllers ist auf der Platine ein 2x3-Pinheader vorgesehen.
Es sind nur wenige SMD-Widerstände und SMD-Kondensatoren zu bestücken. Die beiden 1N4001-Dioden (D2 und D3) werden vom Hersteller für den Betrieb des für 3,3 V gedachten Sound-Moduls an 5 V vorgeschlagen.
Das Sound-Modul kann entweder direkt auf die Platine gelötet oder aber über IC-Sockel gesteckt werden.
An den Ausgang (J9) lassen sich Lautsprecher oder Kopfhörer mit Impedanzen zwischen 8 Ω und 32 Ω anschließen. Die Ausgangsleistung von maximal 250 mW ermöglicht bei passendem Lautsprecher eine unüberhörbare Lautstärke.
Die 3-mm-LED D4 zeigt an, wenn das SOMO-Modul aktiv ist (*busy*). Wenn man das Busy-Signal herausführen möchte, kann man es auf den noch freien Pin 1 von J11 legen.
Die I²C-Verbindung kann über Kabel oder steckbar per SIL-Steckverbindung realisiert werden. Die Pins an J11 sind wie folgt belegt:

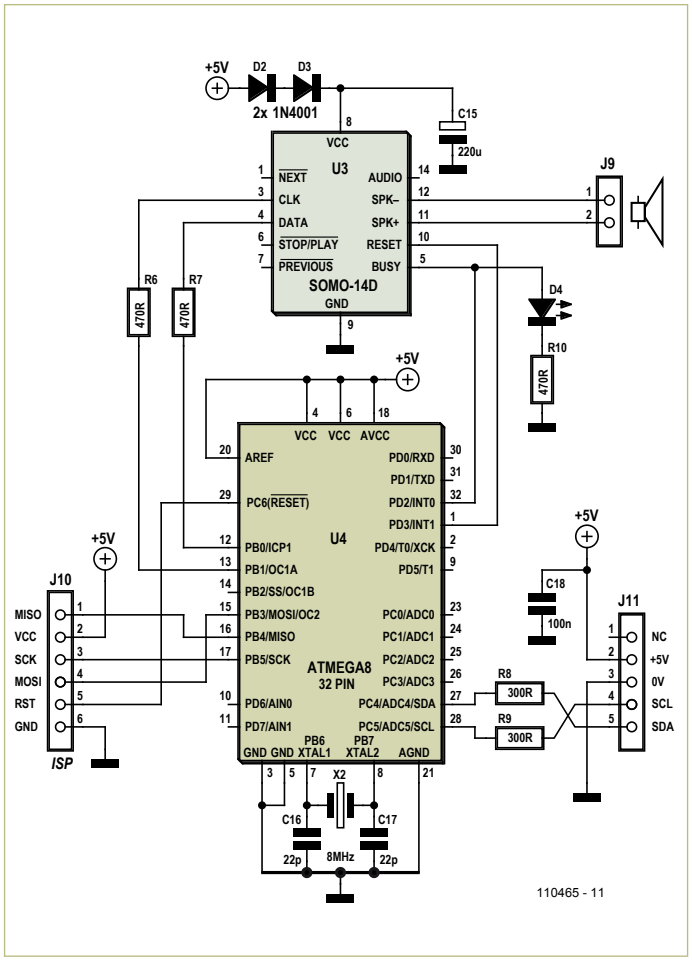


Bild 1. Schaltung der elektronischen Stimme.

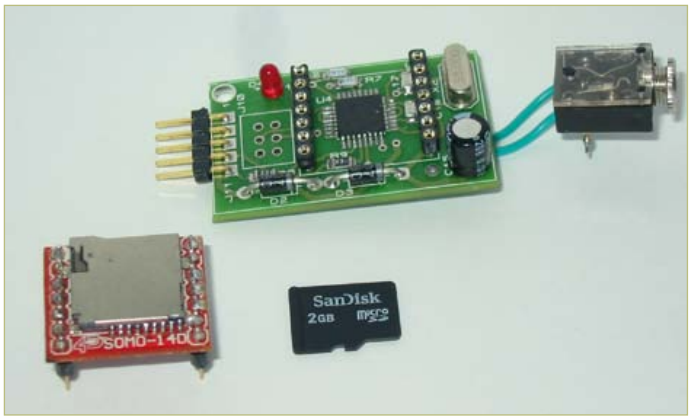
Pin	Funktion
1	nicht benutzt
2	+5 V
3	Masse
4	SCL
5	SDA

Alles auf SD-Karte

Auf der microSD-Karte sind die Dateien für die Zahlen und sonstige Meldungen abgelegt. Da microSD-Karten bis zu 2 GB Kapazität verwendet werden können, ergeben sich vielfältige Einsatzmöglichkeiten. Für die Zahlenausgabe in französischer Sprache sind 105 Dateien erforderlich. Somit bleiben dem Anwender noch 407 Dateien, die er mit beliebigen Inhalten für diverse Einheiten, Wörter, Phrasen, Musik oder Alarmsignale verwenden kann.

I²C-Befehle

Die Modul-Adresse ist $58. Das Zugriffsprotokoll ist nichts Besonderes. START wird gefolgt von der Modul-Adresse. Dann kommen die Daten in Form der ASCII-Zeichen des zu verarbeitenden Strings. Zum Schluss kommt der Befehl STOP.
Der Autor hat nicht alle direkten Befehle ausprobiert. Der Befehl für die Lautstärkeeinstellung funktioniert jedenfalls auch mit direkter Adressierung der Sound-Datei.



Die wichtigsten Komponenten der elektronischen Stimme.

Das fertige Sprach-Modul.

SOMO-14D-Befehle laut Herstellerangaben		
Befehls-Code	**Funktion**	**Beschreibung**
0000h – 01FFh	Nummer der Audio-Datei	Adressiert eine Sound-Datei auf der microSD-Karte. Maximal 512 Dateien sind möglich.
FFF0h – FFF7h	Volume	Code für die Lautstärke mit acht Stufen.
FFFEh	Play/Pause	Wiedergabe oder Pause der aktuellen Datei.
FFFFh	Stop	Stoppt die Wiedergabe der Datei und versetzt das Modul in den Wartezustand.

Datei-Spezifika

Die wichtigsten Dateien für die Ausgabe in französischer Sprache können von [2] herunter geladen werden. Sie sind für die Ablage auf einer SD-Karte vorbereitet. Der Anwender kann allerdings auch seine eigenen Dateien erzeugen und diese Dateien dadurch ersetzen.

Die elementaren Dateien enthalten die Sprachausgabe von „null“ bis „hundert“, „tausend“ (0101d), „mega“ (0144d), „komma“ (0129d), „punkt“ (0130d) und „minus“ (0102d). Die Software ist so gestrickt, dass die Sprach-Dateien für die Zahlen 0...99 in den entsprechenden Dateien mit den Nummern 0...99 abgelegt sind. Die Software verwandelt diese in hexadezimale Zeichen, damit das Sound-Modul korrekt angesteuert wird. Dies gilt für das Vorlesenlassen von Zahlen. Direkte Befehle werden hexadezimal codiert.

Man kann die nötigen Sound-Dateien mit synthetischer Sprache generieren oder einfach gesprochene Worte aufzeichnen. Wichtig ist lediglich, dass die Aufnahmen eine bestimmte Mindestlänge aufweisen, damit sie als gültig gewertet werden. Zu kurze Dateien werden vom SOMO-Modul ignoriert.

Von der Webseite des Herstellers kann man eine Utility (SOMO Tool) [3] herunterladen, mit der man Sound-Dateien des Typs „.wav“ oder „.mp3“ in das hier verwendete Format „.ad4“ konvertieren kann.

Laut Hersteller sind bestimmte microSD-Karten inkompatibel. Der Autor verwendete Karten von SanDisk, die bis heute fehlerfrei arbeiteten.

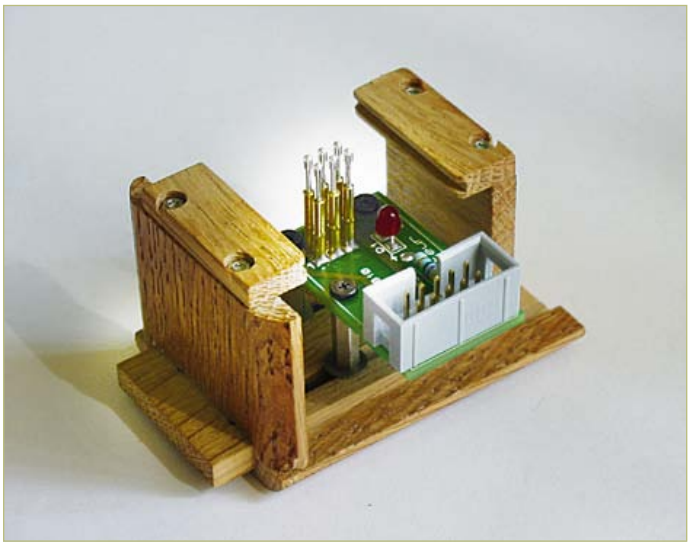
(110465)

Weblinks

[1] SOMO-Modul: www.4dsystems.com.au/prod.php?id=73

[2] Software und Sound-Dateien des Autors (französisch): www.elektor.com/110465

[3] Audio-Konverter für das SOMO-Modul: www.4dsystems.com.au/prod.php?id=74



Bei beengten Platzverhältnissen können die Stifte eines Pinheaders stören. Der Autor hat sich deshalb diese Programmierhilfe einfallen lassen.

Elektor Academy Web-Seminare - in Kooperation mit *element14*

Elektronik-Fans aufgepasst: Elektor Academy und die Community-Plattform element14 präsentieren mehrere (englischsprachige) Webinare zu besonders interessanten Elektor-Projekten. Die Teilnahme ist komplett kostenlos! Sie müssen sich nur unter der Adresse www.elektor.de/webinar registrieren.

Webinar-Programm:

E-Blocks, Twitter and the Sailing Club
Datum: Donnerstag, 17. November 2011
Beginn: 16.00 Uhr MEZ
Präsentiert von: Ben Rowland und John Dobson (Matrix Multimedia)

E-blocks sind kleine Schaltungs-Boards für typische Aufgaben der Elektronik- und Embedded-Entwicklung. In diesem Webinar demonstrieren Ben Rowland und John Dobson die Rapid-Prototyping-Fähigkeiten der E-blocks – am Beispiel einer Applikation, die Twitter-Meldungen an die Mitglieder eines Segelclubs absetzt.

Let's Build a Chaos Generator
Datum: Donnerstag, 15. December 2011
Beginn: 16.00 Uhr MEZ
Präsentiert von: Maarten Ambaum und R. Giles Harrison (Reading University)

Dieses Webinar ist ein Making-Of des interessanten Chaos Generator Projekts, das im September und Oktober 2011 in Elektor vorgestellt wurde. Seien Sie in Wort und Bild dabei, wenn wir die OpAmps zum Glühen bringen!

Here comes The Elektor Bus!
Datum: Donnerstag, 19. Januar 2012
Beginn: 16.00 Uhr MEZ
Präsentiert von: Jens Nickel (Elektor)

Zum ElektorBus-Projekt haben Dutzende Leser mit Ideen, Hinweisen und eigenen Entwicklungen beigetragen. Elektor-Redakteur Jens Nickel erzählt uns, wie alles anfing - doch er kann auch mit vielen Hintergrund-Details zu diesem interessanten Projekt aufwarten, das immer wieder knifflige Herausforderungen bereithält.

Webinar-Archiv:

Platino – an ultra-versatile platform for AVR microcontroller circuits
Präsentiert von: Clemens Valens (Elektor)

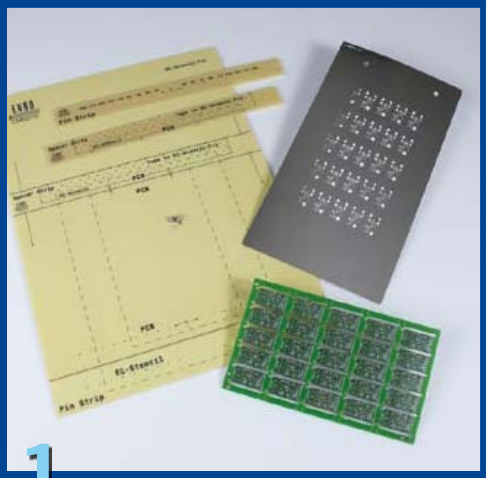
In vielen Elektronik-Projekten kommt die Platine oft nur an zweiter Stelle. Natürlich ist es einfacher, eine Schaltung auf einer Platine aufzubauen. Eine Schaltung lässt sich aber auch ohne Platine realisieren - eine Platine jedoch nicht ohne Schaltung. Dass das nicht immer so sein muss, beweist das Platino-Projekt. Dabei dreht sich alles um die Platine, und die Schaltung ist eigentlich nicht so wichtig.

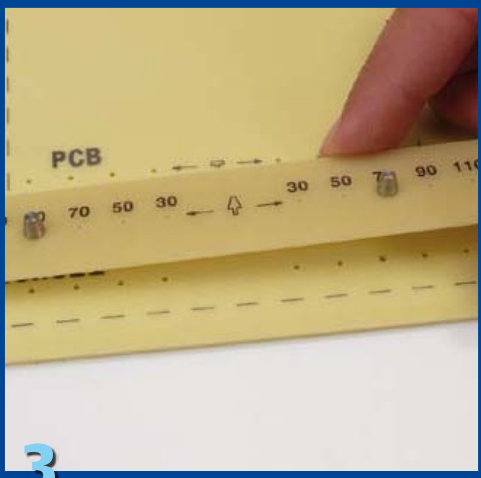
Webinar verpasst? Aufzeichnung abzurufen unter www.element14.com!

elektor ACADEMY
the school of electronics

element14
www.element14.com

Die Teilnehmerzahl ist begrenzt! JETZT ANMELDEN unter www.elektor.de/webinar!





Stencils in der Praxis

Von Thijs Beckers (Redaktion NL / Elektor-Labor) &
Antoine Authier (Elektor-Labor)

Die Platinenherstellung ist für Elektroniker keine Geheimwissenschaft. Doch gibt es Dinge, die etliche Elektroniker vermutlich noch nie im Leben ausprobiert haben. Darunter fällt wohl das Aufbringen von Lötpaste mit Hilfe eines so genannten Stencils. Wenn man bei einem der bekannten Dienstleister eine Platine für die SMD-Bestückung bestellt, bekommt man oft auch die Option zu sehen, ein passendes Stencil mitzubestellen. Ein Stencil ist eine Schablone, die das Aufbringen eben der Lötpaste vereinfacht, verbessert und beschleunigt. Ein gutes Beispiel hierfür ist die Platine zum Elektor USB-FT232R Breakout-Board („BOB“).

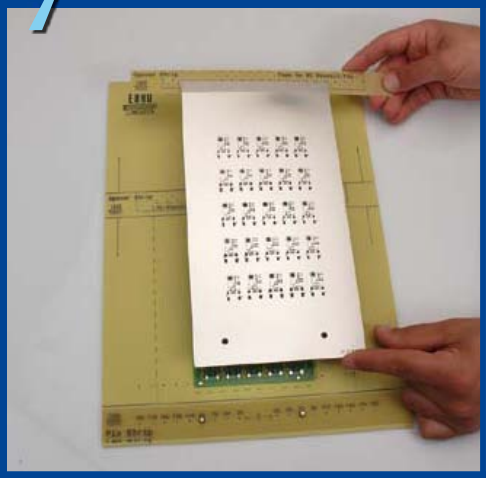
Bild 1
Das sind die Einzelteile, die bei einem Stencil-Kit dabei sind: vorne die Platine, rechts das Stencil und links die Halterplatine mit Befestigungsmaterial.

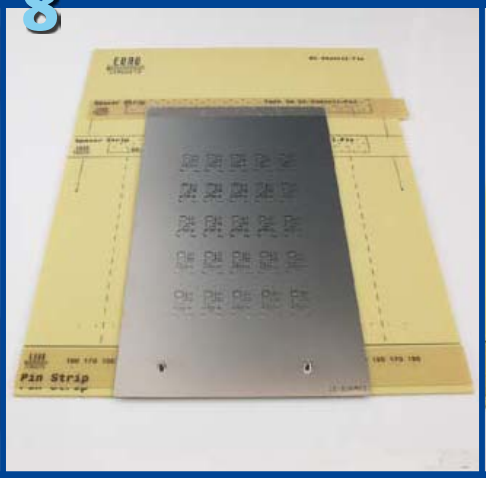
Bild 2
Mit diesen speziell geformten Pins werden Stencil und Platine genau fixiert.

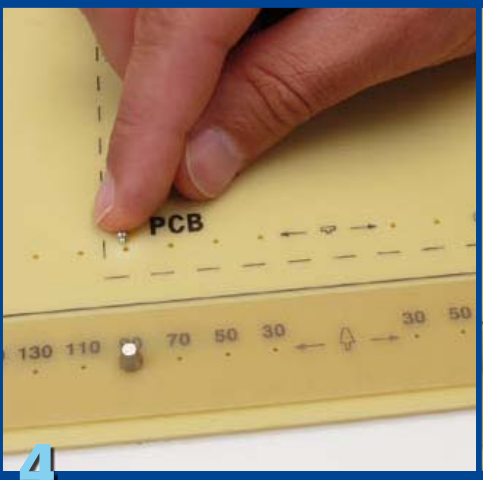
Bild 3
Hierzu steckt man die kegelförmigen Stifte auf den Streifen Platinenmaterial, der dann auf die Halterplatine kommt.

Bild 4
Dann kommen die Stifte für die Platinenfixierung auf die Halterplatine.

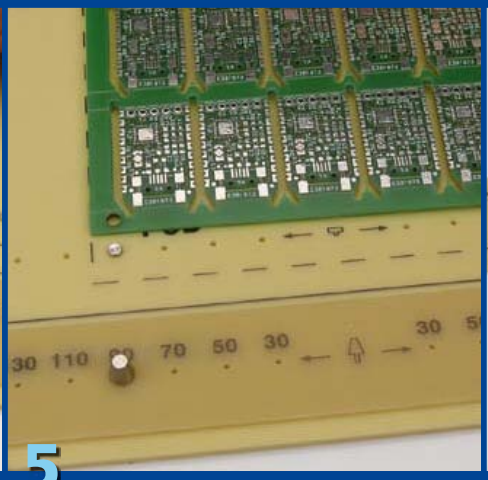
Bild 5
Nun kann die Platine auf der Halterplatine befestigt werden. Die Platine muss sauber sein. Fingerabdrücke und Verunreinigungen auf den Lötpads sollten vermieden werden.







4



5

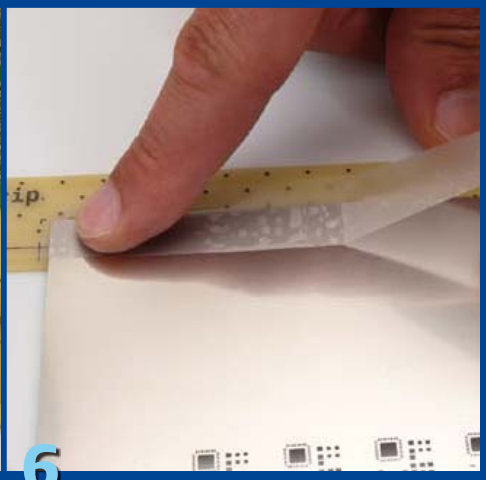

6

Bild 6
Nun muss man dafür sorgen, dass das Stencil glatt und ohne Verbiegung auf der Platine aufliegen kann. Hierfür klebt man das Stencil an der oberen Kante auf ein Stück Platine, das die gleiche Stärke wie die Nutzplatine aufweist.

Bild 7
Jetzt wird das Stencil auf die Platine gelegt.

Bild 8
Die kegelförmigen Stifte halten das Stencil sehr präzise am vorgesehenen Platz. Man sieht, dass die Lötpads frei liegen.

Bild 9
Die Lötpaste kann nun aufgebracht werden. Es wird zunächst nur die erste Reihe der Einzel-Platinchen mit Lötpaste versorgt, weshalb man nicht zu viel davon auftragen sollte.

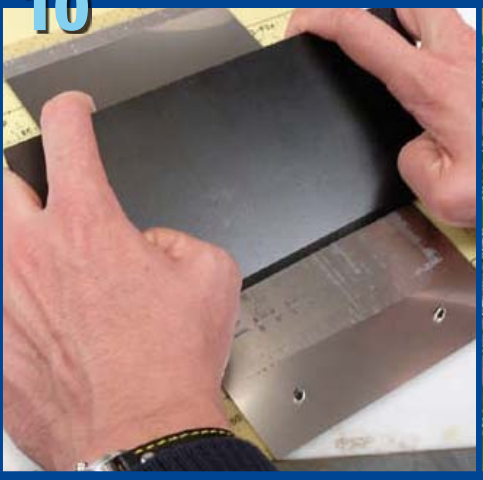
Bild 10
Dann muss man die Lötpaste mit einer einzigen Bewegung per Rakel über das Stencil ziehen, um sie zu verteilen. Für beste Resultate bewegt man den Rakel auf sich zu.

Bild 11
Nach dem Entfernen des Stencils überprüft man, ob die Lötpaste perfekt aufgebracht wurde.

Bild 12
In der Nahaufnahme kann man sehen, dass selbst die kleinsten Pads für IC-Pins die richtige Dosis Lötpaste abbekommen haben – beachtenswert, denn der Pin-Abstand beträgt hier nur 0,5 mm!

Die Platine ist nun fertig für die Bestückung mit Bauteilen. Wenn diese platziert sind, geht es ab in den Lötofen.

(110514)


10


11


12

Low-Cost-DMX-Mischpult

Farbsteuerung für LED-Spot

Bild 1. Der vom Autor verwendete PAR56-RGB-LED-Scheinwerfer mit 153 LEDs. Sehr häufig sind auch ähnliche Ausführungen wie zum Beispiel PAR64-RGB-LED-Scheinwerfer mit 183 LEDs.

Von Ingo Sundermann (D)

Auf der Suche nach einer kostengünstigen Ansteuerung für einen preiswerten LED-Spot mit 153 LEDs und DMX-Interface entstand dieses Mini-Lichtmischpult als Eigenentwicklung. Mit drei Schiebepotis an einem ATmega8 lässt sich die Intensität der Farben Rot, Grün und Blau einstellen.

LED-Spots für die Bühnen- und Partybeleuchtung werden heutzutage schon recht günstig angeboten. Meist handelt es sich um so genannte PAR-Scheinwerfer (Parabolic Aluminized Reflector) mit einer standardisierten zylindrischen Bauform (so genannte „Can“). Die verschiedenen Größen sind durch den Durchmesser in Achtelzoll und zwei verschiedene Längen („longnose“ oder „shortnose“) definiert. So hat zum Beispiel ein PAR56-Scheinwerfer einen Can-Durchmesser von etwa 18 cm. Wenn keine sehr hohe Lichtleistung gefordert ist, lassen sich LED-bestückte PAR-Cans sehr vorteilhaft einsetzen, da sie auf Grund des hohen Wirkungsgrads weniger Abwärme produzieren und eine höhere Lebensdauer aufweisen. Bei Bestückung mit roten, grünen und blauen LEDs ist außerdem sehr einfach eine RGB-Steuerung der Lichtfarbe über DMX möglich. DMX (**D**igital **M**ultiple**X**) ist ein in der Bühnen- und Veranstaltungstechnik als Standard etabliertes Steuerprotokoll, das zur Übertragung den RS485-Bus verwendet. Von einem Master (normalerweise ein DMX-Steuer- oder Mischpult) aus können in einer Nachricht 1-Byte-Daten für bis zu 512 Kanäle (Funktionen) versandt werden.

Ausgangspunkt für das hier vorgestellte Leserprojekt war ein preisgünstiger PAR56-LED-Scheinwerfer mit 153 LEDs (**Bild 1**). Da die automatische „Music to Light-Erkennung“ bei diesem Exemplar nicht besonders zufriedenstellend war, wurde ein einfaches RGB-DMX-Mischpult gesucht – und selbst entwickelt, weil nichts Passendes zu finden war.

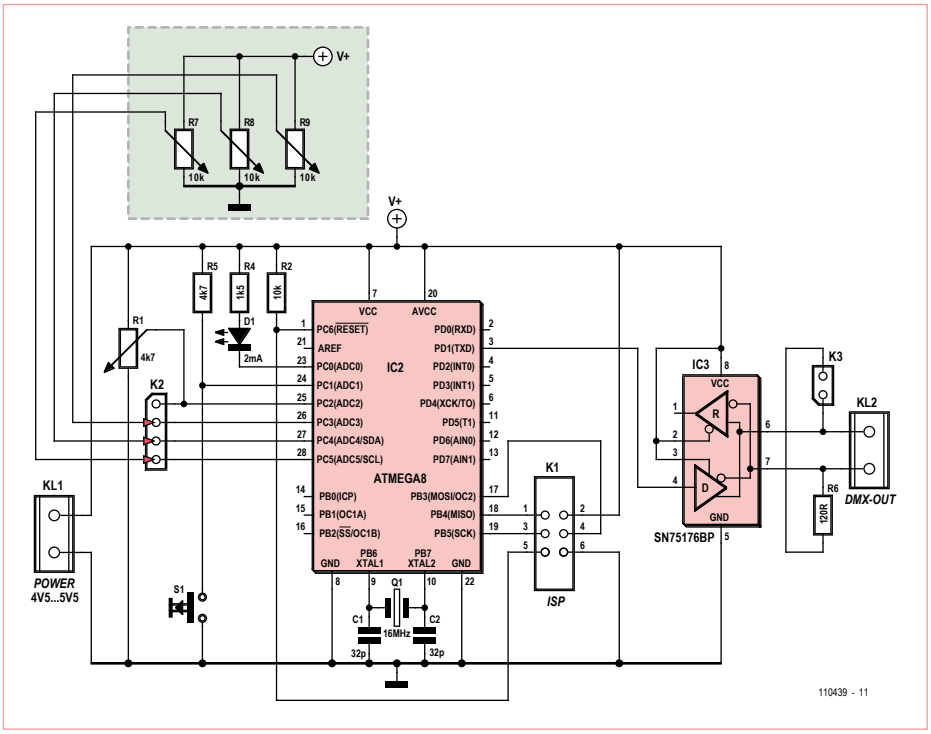


Bild 2. Die Schaltung besteht im Wesentlichen aus einem ATmega8-Mikrocontroller und einem RS485-Transceiver.

Hardware

Für die relativ einfache Anforderung, die Farben Rot, Grün und Blau zu mischen, wäre ein professionelles Lichtmischpult zwar geeignet, aber viel zu teuer. Greift man zu einem ATmega und entwickelt selbst, liegen die Bauteilkosten mit Standard-Potis unter 10 Euro. Außerdem besteht die Möglichkeit,

Leserprojekte sind Beiträge von Elektor-Lesern für experimentelle Zwecke oder zur Anregung für andere Leser.
Die in dieser Rubrik vorgestellten Schaltungen wurden vom Elektor-Labor nicht auf Reproduzierbarkeit und Funktion getestet.

bei Bedarf Erweiterungen hinzuzufügen, da sich die Software leicht modifizieren lässt.
Als Mikrocontroller wird ein ATmega8 eingesetzt. Die darauf laufende Software umfasst das Einlesen der drei Analogwerte für die Farben rot, grün und blau und das Versenden der Daten über den DMX-Bus.
Die Schaltung in **Bild 2** besteht im Wesentlichen aus dem Mikrocontroller (IC1), einem RS485-Transceiver (IC2) und den Potis. Da für DMX eine Baudrate von 250 kBaud spezifiziert ist, benötigt der ATmega-Taktoszillator einen 16-MHz-Quarz (X1). Der RS485-Tranceiver ist an die TX-Leitung der seriellen Schnittstelle des ATmega angeschlossen. Da die RS485-Busleitung an den beiden Enden mit 120 Ω abgeschlossen („terminiert“) werden muss, ist am RS485-Anschluss der Schaltung ein 120-Ω-Widerstand mittels Jumper zuschaltbar.
Die Spannungen von den Schleifern der Potis sind zur A/D-Wandlung an Port C angeschlossen. An PC3..PC5 liegen Spannungen der Schieberegler für die RGB-Steuerung (Dimmung). Diese Helligkeitspotis müssen einen logarithmischen Verlauf aufweisen, da das menschliche Auge die Helligkeit nicht linear wahrnimmt. Um den Einstellbereich für die Dimmung zu optimieren, kann man den RGB-Potis einen Widerstand vorschalten. Das vierte Poti (an PC2) dient zur Einstellung der Frequenz des Stroboskopeffekts und sollte daher einen linearen Verlauf haben. Mit Hilfe des Tasters können bei Bedarf Zusatzfunktionen aktiviert werden. In der aktuellen Software-Version dient er zum Einschalten des Stroboskop-Effekts.
Die Spannungsversorgung der Schaltung erfolgt durch vier NiMH-Akkus der Größe AA (Mignon) mit einer Spannung von typisch 4,8 V (ca. 5,3 V bei voll geladenen Akkus). Bei einer Stromaufnahme der Schaltung von etwa 60 mA ergibt sich bei 2500 mAh Akkukapazität eine beachtliche Betriebsdauer von über 40 Stunden. Selbstverständlich kann man bei stationärem Betrieb anstelle der Akkus auch ein stabilisiertes 5-V-Netzteil (100 mA) anschließen.
Für den Aufbau der Schaltung wurde eine Platine entwickelt (**Bild 3**). Auf der Webseite zu diesem Projekt [1] steht das Platinenlayout im PDF- und T3001-Format (für TARGET 3001!) zum kostenlosen Download zur Verfügung.
Beim Musteraufbau (siehe **Bild 4**) wurden die drei Schiebepotis für die RGB-Einstellung auf einer separaten Platine untergebracht.

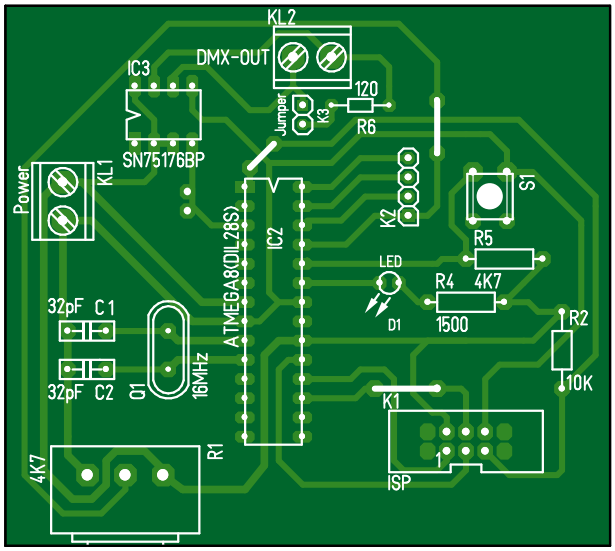


Bild 3. Der Bestückungsplan der Platine, die sich sehr gut selbst herstellen lässt.

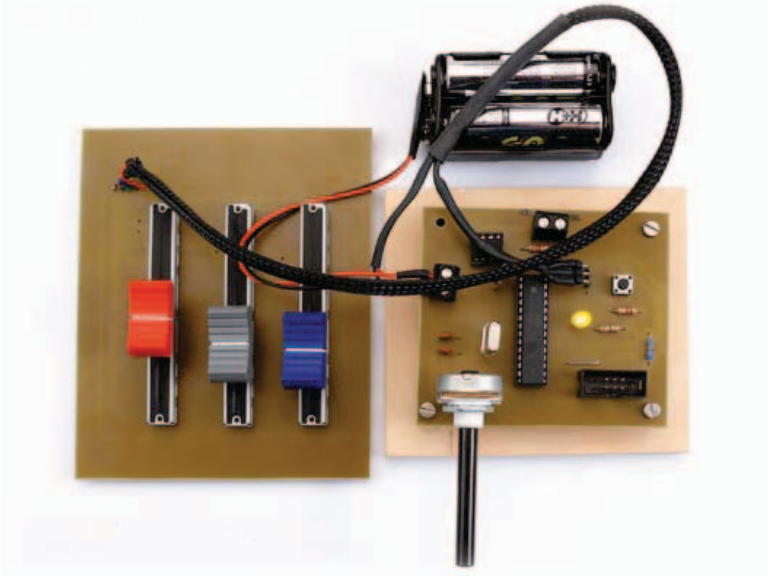


Bild 4. Beim Musteraufbau hat der Autor die drei Schiebepotis für die RGB-Einstellung auf einer separaten Platine untergebracht.

Software

In der in C programmierten Firmware laufen zwei Zustandsmaschinen. Zustandsmaschine 1 regelt die DMX-Kommunikation und Zustandsmaschine 2 die Analog-Digital-Konvertierungen.
Für die DMX-Kommunikation werden im aktuellen Projekt nur fünf DMX-Kanäle verwendet. Für den angeschlossen LED-Scheinwerfer reicht dies aus. Das DMX-Protokoll ist aus dem Diagramm in **Bild 5** zu erkennen.
Für den Reset-Impuls (BREAK) wird die serielle Schnittstelle kurzzeitig deaktiviert und der Pin PD1 auf Low gezogen. Die darauf folgenden DMX-Kanäle können ohne Probleme erweitert werden. Die Startadresse am Scheinwerfer ist auf Kanal 1 eingestellt. Damit entspricht der Dimmkanal für Rot dem Kanal 1. Hier die Kanalbelegung für die Farbdimmung:

Rot	DMX-Kanal 1
Grün	DMX-Kanal 2
Blau	DMX-Kanal 3

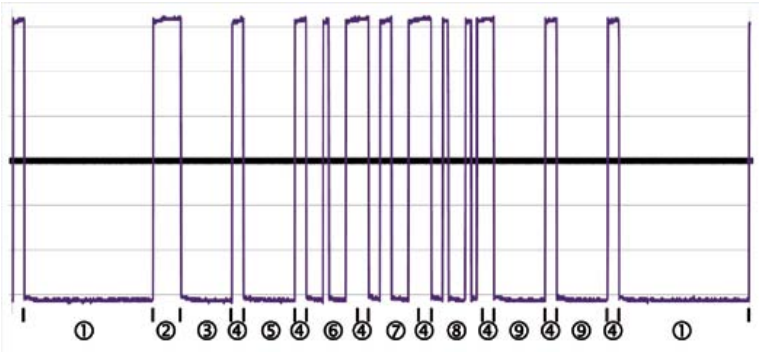


Bild 5. Das DMX-Timing-Diagramm zeigt einen Datenblock, der sich aus folgenden Abschnitten zusammensetzt:

1 - BREAK (91 µs)
2 - Mark after BREAK (20 µs)
3 - START (Inhalt muss 0x00 sein) START-Bit (4 µs) + 8 Datenbits (32 µs)
4 - 2 STOP-Bits (8 µs)
5 - DMX-Channel 0 START-Bit (4 µs) + 8 Datenbits (32 µs)
6 - DMX-Channel 1 (Dimmstufe rot) START-Bit (4 µs) + 8 Datenbits (32 µs)
7 - DMX-Channel 2 (Dimmstufe grün) START-Bit (4 µs) + 8 Datenbits (32 µs)
8 - DMX-Channel 3 (Dimmstufe blau) START-Bit (4 µs) + 8 Datenbits (32 µs)
9 - DMX-Channel 4 und 5 (Inhalt 0x00) START-Bit (4 µs) + 8 Datenbits (32 µs)
Die Reihenfolge der Datenbits ist LSB...MSB.

Die vier A/D-Kanäle werden nacheinander ausgelesen: PC3 bis PC5 für die RGB-Dimmwerte und PC2 (geplant) für die Blinkfrequenz (Stroboskop-Wert). Die 10-bit-Werte werden durch Shiften auf 8-bit-Werte reduziert, weil im DMX-Format nur der Wertebereich von 0...255 übertragen wird.

Zur Programmierung des Mikrocontrollers ist auf der Platine ein ISP-Anschluss (2x5-polige Stiftleiste) vorgesehen. Die Firmware (Source- und Hexcode) kann von der Webseite [1] zu diesem Projekt kostenlos heruntergeladen werden.

Kaskadierung

Da DMX [2] den RS485-Bus für die Übertragung verwendet, lassen sich bis zu 32 Empfänger parallel an den Bus anschließen. Die LED-Scheinwerfer verfügen zum Durchschleifen der Busleitung über einen DMX-Eingang (3-poliger XLR-Einbaustecker) und einen DMX-Ausgang in Form einer 3-poligen XLR-Buchse (**Bild 6**). Wie bereits erwähnt, muss die Busleitung am Anfang und am Ende mit einem 120-Ω-Widerstand abgeschlossen werden. Auf der Platine des DMX-Mischpults ist dieser Widerstand am DMX-Ausgang bereits vorgesehen und über einen Jumper zuschaltbar. Für den Bus-Abschluss am Ende der Buslinie kann man einfach einen XLR-Stecker mit Widerstand (siehe **Bild 7**) verwenden, der in die DMX-Ausgangsbuchse des letzten Scheinwerfers gesteckt wird.

Erweiterungen

In der aktuellen Version der Software wird der Stroboskopeffekt mit dem Taster S1 ein- und ausgeschaltet, wobei die Stroboskopfrequenz noch nicht über DMX eingestellt wird. Dafür könnte der Spannungswert vom Schleifer des linearen Potis an PC2 eingelesen und ausgewertet werden. Zusätzlich sollte dann ein Timer-Interrupt integriert werden, um die Frequenz des Strobos von Programmlaufzeiten unabhängig zu machen.
Eine interessante Erweitung könnte ein automatischer Ablauf von Farbprogrammen sein. Eine weitere Möglichkeit wäre die Implementierung einer DCF-77-Uhr, um die Farbabläufe abhängig von Datum und Uhrzeit zu steuern. So bald es eine Software-Version gibt, wird diese auf der Projekt-Webseite [1] zum kostenlosen Download bereitgestellt.

(110439)

Weblinks:

[1] www.elektor.de/110439 (Webseite zu diesem Projekt)
[2] www.elektor.de/010035 (Artikel „DMX512“ aus Elektor 5/2001)

Bild 6. DMX-Ein- und Ausgang auf der Rückseite des RGB-LED-Scheinwerfers. Die DIP-Schalter dienen im DMX-Betrieb zur Einstellung des DMX-Kanals.

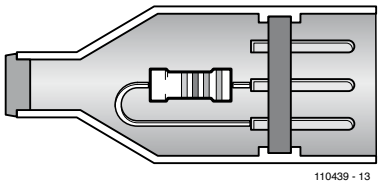


Bild 7. Ein XLR-Stecker mit eingebautem 120-Ω-Widerstand muss für den Bus-Abschluss in die DMX-Ausgangsbuchse des letzten Scheinwerfers einer Buslinie gesteckt werden.

Der Autor

Nach seiner Ausbildung zum Tontechniker hat Ingo Sundermann an der Fachhochschule Köln (Abteilung Gummersbach) E-Technik (Fachrichtung Industrieelektronik) studiert.

Seit Abschluss des Studiums ist er als Entwicklungsingenieur tätig, zurzeit (und mit Begeisterung) im Bereich der Fahrzeugdiagnose für mobile Baumaschinen. Sein besonderes Interesse gilt schon seit langem der Realisierung von Schaltungen mit und für LEDs.

elektor

Elektor Print

Gewohnter Lesespaß auf Papier

+

Elektor Digital

Neuer Lesespaß auf PC, Notebook oder Tablet

=

Elektor PLUS

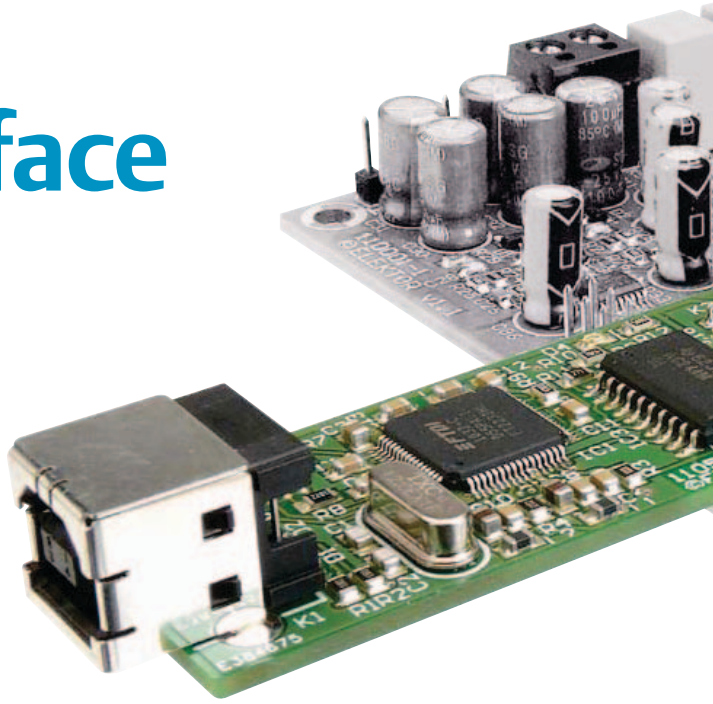
Ultimativer Lesespaß zu Hause oder unterwegs

Lesen Sie Elektor im vorteilhaften PLUS-Abonnement!

Jetzt abonnieren oder upgraden: www.elektor.de/abo

OnCE/JTAG-Interface

Zum Programmieren und Debuggen von Freescale-DSPs



Von Ton Giesberts (Elektor-Labor)

Das DSP-Board aus unserem DSP-Kurs wird durch einen USB-Adapter ergänzt, der eine schnelle und technisch zeitgemäße Verbindung zum PC herstellt. Über den Adapter lässt sich das DSP-Board am PC-Bildschirm programmieren und debuggen. Der Adapter passt auch zu anderen DSP-Typen aus der DSP56K-Familie von Freescale.

Im Fokus des für unseren DSP-Kurs entwickelten DSP-Boards, das wir in der September-Ausgabe vorgestellt haben, steht der DSP56374 aus der Symphony-Reihe von Freescale. Programmiert wird der DSP über eine JTAG-Schnittstelle, für die Freescale einen „OnCE“ („On-Chip Emulation“) genannten eigenen 14-poligen Steckverbinder verwendet. Aus Platzgründen ist auf dem Board keine direkte Schnittstelle für die Kopplung mit einem PC vorhanden. Zwar bietet Freescale mehrere geeignete Programmieradapter an, doch sie sind vergleichsweise teuer. Wer schon einmal mit einem Entwicklungskit von Freescale gearbeitet hat, ist vermutlich bereits im Besitz eines solchen Programmieradapters. Im Internet werden eine Reihe mehr oder weniger einfache Selbstbau-Lösungen für die Programmierung der Freescale-DSPs beschrieben. Leider benutzen die meisten Vorschläge den PC-Parallelport, der an modernen PCs kaum noch zu finden ist. Deshalb folgt hier ergänzend zum DSP-Board ein OnCE/JTAG-Adapter mit USB-Anbindung, der für eine schnelle und zeitgemäße Kopplung zwischen DSP-Board und PC sorgt.

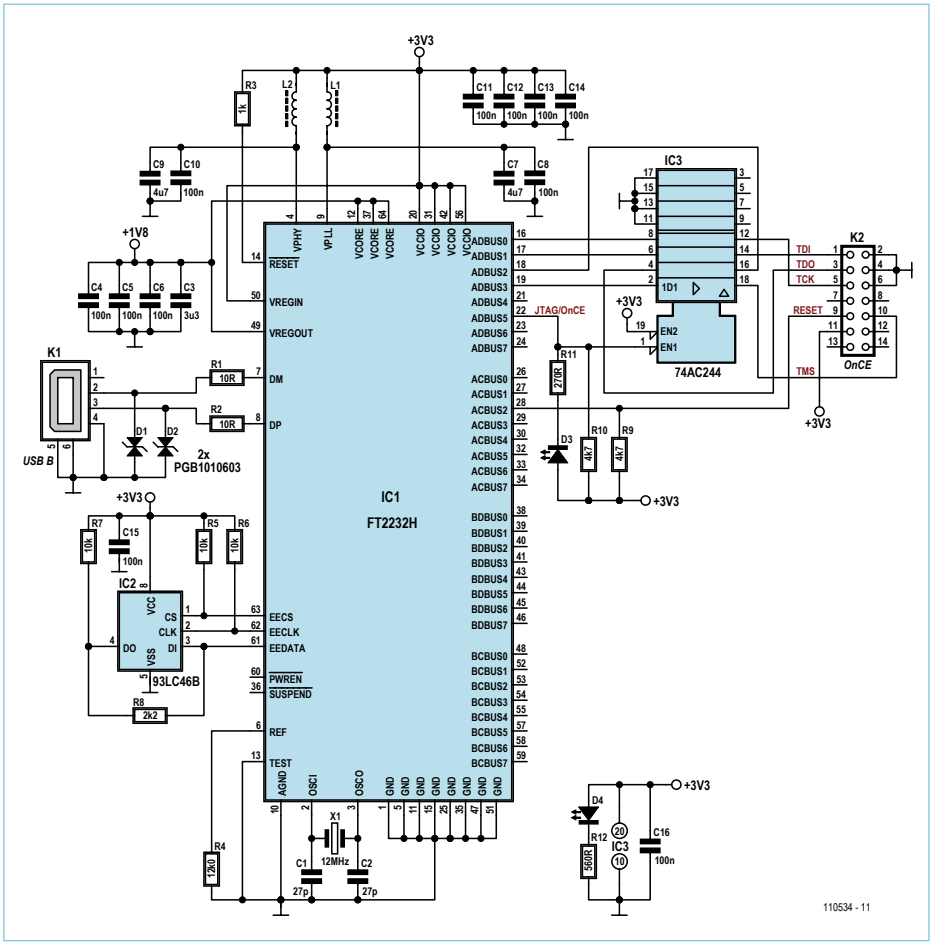
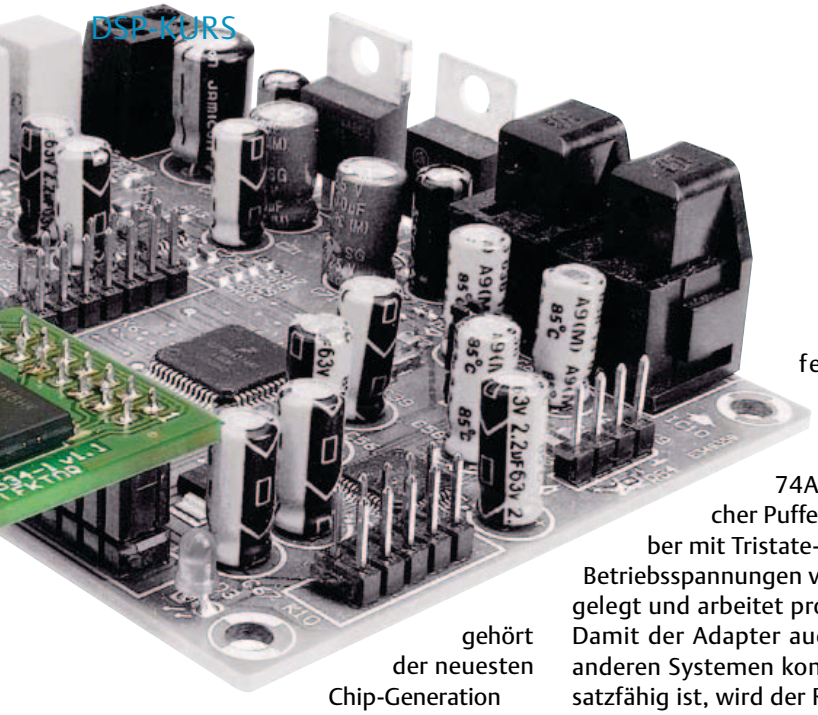


Bild 1. Der FT2232H von FTDI ist der zentrale Baustein des Programmieradapters für das DSP-Board.

Adapter-Hardware

Für unseren OnCE/JTAG-nach-USB-Adapter haben wir den *High-Speed USB UART/FIFO* mit der Typenbezeichnung FT2232H von FTDI gewählt (IC1 in **Bild 1)**. Dieser Baustein



gehört der neuesten Chip-Generation von FTDI an, was sich auch in den umfangreichen Dokumentationen [1] widerspiegelt. Über diesen Chip kann das DSP-Board in USB-2.0-Geschwindigkeit mit dem PC kommunizieren. Die Betriebsspannung von 3,3 V liefert das DSP-Board, denn diese Spannung ist dort bereits vorhanden. Der Kern des FT2232H arbeitet mit der niedrigeren Spannung 1,8 V, diese Spannung stellt ein im Baustein integrierter Spannungsregler bereit. Für die Verbindung mit einem USB-Port des PCs werden lediglich zwei Widerstände sowie die USB-B-Buchse K1 benötigt. Die ESD-Suppressor-Dioden D1 und D2 schützen den USB-Eingang zusätzlich vor den Auswirkungen elektrostatischer Ladungen. Die Schaltzeiten dieser Dioden betragen weniger als 1 ns, dank der extrem niedrigen Kapazität 0,055 pF haben die Dioden keinen Einfluss auf die USB-Signale. Vom Reset (aktiv: Low) des FT2232H wird hier kein Gebrauch gemacht, der Reset-Anschluss liegt über R3 an der Betriebsspannung. Das externe EEPROM IC2 ist in Standard-Konfiguration über R5...R8 mit dem FT2232H verbunden. Dieses EEPROM muss ein Typ mit 16 bit Wortbreite und 3,3 V sein. Eine Speicherkonfiguration (der zugehörige Anschluss wird üblicherweise mit „ORG“ bezeichnet) ist nicht erforderlich. Für unsere Zwecke ist der 93LC46B/SN von Microchip gut geeignet. Die Kondensatoren C3...C14 sorgen zusammen mit den Induktivitäten L1 und L2 für eine wirksame Entkopplung der Betriebsspannung.

Wir haben uns entschlossen, die Ausgangssignale (mit Ausnahme des Reset-Signals) über Puffer herauszuführen. Solange der Adapter in der aktuellen Entwicklungsumgebung nicht in Betrieb ist, sind die gepufferten Ausgänge deaktiviert und somit hochohmig. Der schnelle 74AC244, ein achtfacher Puffer und Leitungstreiber mit Tristate-Ausgängen, ist für Betriebsspannungen von 1,5...5,5 V ausgelegt und arbeitet problemlos an 3,3 V. Damit der Adapter auch zusammen mit anderen Systemen komplikationslos einsatzfähig ist, wird der Reset-Ausgang mit Pullup-Widerstand R9 auf +3,3 V hoch gezogen. Auf dem DSP-Board befindet sich bereits ein Pullup-Widerstand, bei ausschließlichem Betrieb an diesem Board kann R9 entfallen.

Bild 2 zeigt die für den Adapter entworfene, mit SMD-Bauelementen bestückte Platine.

Die 2 x 7-Pin-Buchsenleiste hat ihren Platz auf der Platinenrückseite, so dass der Adapter auf die Stiftleiste K8 des DSP-Boards aufgesteckt werden kann. Im Elektor-Shop ist außer der leeren Platine auch eine bereits bestückte, programmierte und getestete Version erhältlich [2].

Software für Symphony Studio

Wir sind davon ausgegangen, dass der Adapter zusammen mit der Entwicklungsumgebung *Symphony Studio* von Freescale zum Einsatz kommt. Anhand einer Vorgabekonfiguration von Freescale lässt sich der FT2232H so programmieren, dass dieser Baustein als *Symphony SoundBite* erkannt wird. Die aufgebaute und getestete Platine, die im Elektor-Shop erhältlich ist, wurde bereits in dieser Weise programmiert, sie kann ohne weitere Vorbereitungen auf das

Stückliste

Widerstände (SMD 0805, 0,1 W):
R1,R2 = 10 Ω, 5 %
R3 = 1 k, 5 %
R4 = 12k0, 1 %
R5,R6,R7 = 10 k, 5 %
R8 = 2k2, 5 %
R9,R10 = 4k7, 5 %
R11 = 270 Ω, 5 %
R12 = 560 Ω, 5 %

Kondensatoren (SMD 0805):
C1,C2 = 27 p/50 V, 5 %, NP0
C3 = 3µ3/10 V, 10 %, X5R
C4...C6,C8, C10...C16 = 100 n/50 V, 10 %, X7R
C7,C9 = 4µ7/6,3 V, 10 %, X5R

Induktivitäten (SMD 0805):
L1,L2 = 600 Ω @ 100 MHz, 200 mA/0,35 Ω (z.B. Murata BLM21BD601SN1D)

Halbleiter:
D1,D2 = PGB1010603, $V_{clamping}$ = 150 V (Littelfuse, SMD 0603)
D3 = LED grün (Kingbright KPHCM-2012CGCK, SMD 0805)
D4 = LED rot (Kingbright KPHCM-2012SURCK, SMD 0805)
IC1 = FT2232HL-R (FTDI, SMD 64-Pin LQFP)
IC2 = 93LC46B/SN (Microchip, SMD SO-8)
IC3 = CD74AC244M (Texas Instruments, SMD SO-20)

Außerdem:
K1 = USB-B-Buchse, gewinkelt für Platinenmontage
K2 = Buchsenleiste 2 · 7 Kontakte, Raster 2,54 mm
X1 = Quarz 12 MHz, C_{load} 18 pF ±30 ppm, HC-49S
Platine 110534-1

Im Elektor-Shop ist die aufgebaute, programmierte und getestete Platine erhältlich (Bestellnummer 110534-91).

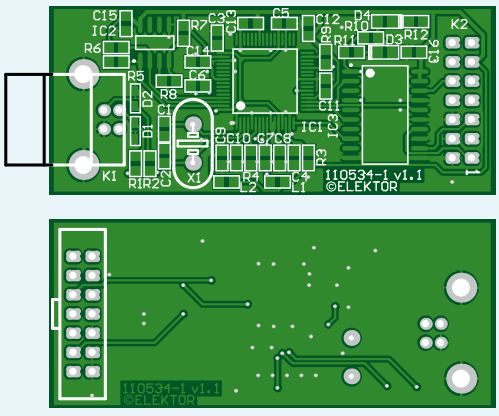


Bild 2. Die Adapterplatine wurde so gestaltet, dass sie auf Steckleiste K8 des DSP-Boards aufgesteckt werden kann.

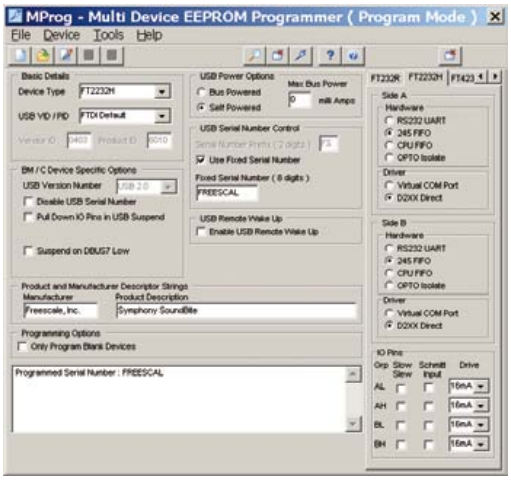


Bild 3. Mit dem Tool „MProg", das FTDI zum Download anbietet, kann der FT2232H programmiert werden.

DSP-Board aufgesteckt werden. In eigner Regie ist der FT2232H mit dem von FTDI bereit gestellten Tool *MProg* programmierbar, die Vorgabekonfiguration von Freescale ist auf dieses Tool abgestimmt. FTDI bietet auch den Nachfolger *FT_Prog* an, dieses neue Tool ist jedoch mit der Vorgabekonfiguration von Freescale nicht kompatibel. Auf der Website von FTDI [3] wird die letzte Version 3.5 von *MProg* auch weiterhin zum Download angeboten.

Leider lässt sich die Single-Channel-Version FT232H mit *MProg* nicht programmieren. Die Dual-Channel-Version ist zwar etwas teurer, doch die Programmierung gestaltet sich einfacher, das Erstellen einer neuen Konfiguration entfällt. Die originale *SoundBite*-Vorgabekonfiguration haben wir in einigen Details angepasst (siehe **Bild 3**), mit der angepassten Version kann die Programmierung sofort starten. Nach einem *Scan* (unter *Device*) erscheinen im Fenster links unten diverse Informationen. Dort ist nachzulesen, ob der Chip vom Programm erkannt wurde, ob er tatsächlich noch nicht programmiert ist und ob gegebenenfalls noch weitere Chips gefunden wurden. Wenn ein leerer FT2232 gefunden wurde, ist er sofort programmierbar, anderenfalls muss er zuvor mit *Erase* gelöscht werden. Für den Selbstbau und die Programmierung des FT2232H in eigener Regie schlagen wir folgende Schritte vor:

Öffnen Sie über das Menü *File, Open* die Vorgabekonfiguration 110534-1.ept, sie steht auf unserer Projektseite [2] gepackt (110534-11.zip) zum Download bereit. Wenn Sie eine Vorgabekonfiguration ändern möchten, müssen Sie nach dem Öffnen unter *File* die Option *Edit* wählen. In der originalen Vorgabekonfiguration von Freescale haben wir folgende Änderungen vorgenommen: Anstelle des Typs FT2232D haben wir den FT2232H als *Device Type* gewählt, und bei den *USB Power Options* haben wir *Self Powered* eingestellt. Nach Auswahl des FT2232H erscheint rechts ein Tab mit den Einstellungen der I/O-Leitungen. Für die Hardware muss die Einstellung von *RS232 UART* in *245 FIFO* geändert werden. Die Treiber sind bereits passend auf *D2XX Direct* eingestellt, diese Treiber [4] müssen installiert sein. Die Dokumentation des FT2232H enthält zwei Anwendungsbeispiele für die Option *Self Powered*, in denen die Busspannung 5 V über Spannungsteiler erkannt wird. Dort ist angemerkt, dass in *MProg* die Option *Suspend on DBUS7 low* eingestellt werden muss. Diese Funktion ist nur verfügbar, wenn der

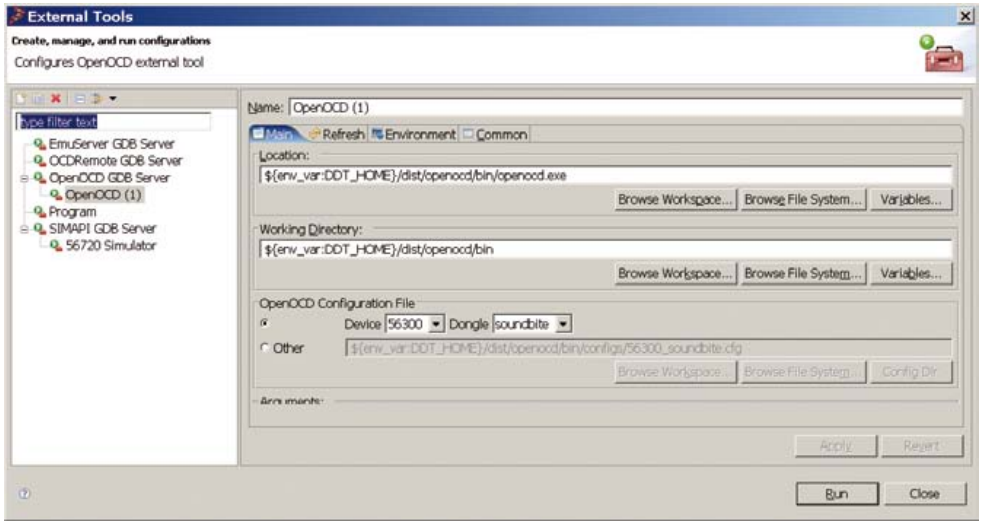


Bild 4. Dieses Fenster des Symphony Studios zeigt, wie der Adapter ausgewählt wird.

Stromaufnahme

Wir haben den Strombedarf unseres Musteraufbaus unter typischen Bedingungen gemessen. Wenn das DSP-Board ohne Adapter betrieben wird (+5 V des digitalen und analogen Teils sind über eine Drossel miteinander verbunden), liegt die Stromaufnahme bei 84 mA. Zusammen mit dem Adapter steigt sie auf 87 mA unmittelbar nach dem Einschalten an. In dieser Phase befindet sich der FT2232H im Suspend-Modus, er begnügt sich mit einigen hundert µA.

Wird der Adapter mit dem PC verbunden, steigt die Stromaufnahme abhängig von der USB-Geschwindigkeit auf 135 mA (bei Full Speed) oder 155 mA (bei High Speed). Nach Konfigurieren des Adapters im Symphonie Studio haben wir 160 mA gemessen. Für den Anstieg um 5 mA ist die grüne LED verantwortlich, sie signalisiert, dass der OnCE-Ausgang aktiv ist. Wenn ein Testprogramm wie „tst_src2.asm" Probe läuft, liegt die Stromaufnahme (mit aktiviertem SRC und digitalen optischen Audio-Signalen) bei ungefähr 270 mA.

zugehörige Spannungsteiler vorhanden ist. Falls die Option versehentlich für diesen Adapter aktiviert wird, kann der PC den Adapter nicht erkennen. Um das Problem zu umgehen, kann das Verbinden von Pin 46 (BDBUS7) mit +3,3 V eine Lösung sein, nicht Pin 59 wie in der Dokumentation angegeben. Als Brücke genügt ein kurzes Stück lackierter Kupferdraht, ∅ 0,1 mm. Kurzschlüsse an den im 0,5-mm-Grid angeordneten Pins müssen natürlich vermieden werden. Alle I/O-Leitungen sind auf den höchsten Stromentnahmewert von 16 mA eingestellt. Durch die grüne LED (D3) fließen ungefähr 4,7 mA, während der Strom durch die rote LED zur Helligkeitsangleichung nur etwa 2,7 mA beträgt. Geänderte Vorgabekonfigurationen müssen gespeichert werden, bevor sie programmiert werden können. Es ist ratsam, das Original nicht zu überschreiben und für die geänderte Version unter *File, Save As* einen anderen Dateinamen zu vergeben. Das Programmieren einer Vorgabekonfiguration ist sofort nach dem Öffnen der Datei möglich.

In der Konfiguration des *Symphony Studio* muss der Adapter als *SoundBite* gehandhabt werden. Der zutreffende Weg, beispielsweise in *C/C++ Perspective*, führt über *Run, External Tools,* noch einmal *External Tools, OpenOCD GDB Server* (beim ersten Mal doppelklicken). Im Tab *Main* muss bei *OpenOCD Configuration File* als Device *56300* und für Dongle *SoundBite* gewählt werden (siehe **Bild 4**).
Wie beschrieben hat dieser Schnittstellenadapter die Form eines Aufsteckmoduls. Steckverbinder K2, der die Verbindung zum DSP-Board herstellt, ist eine auf der Platinenunterseite montierte Buchsenleiste. Für die Verbindung von der USB-Buchse zum PC wird ein Kabel der USB-Spezifikation 2.0 empfohlen, der Kabeltyp ist meistens auf dem Kabel aufgedruckt.

(110534)gd

Weblinks

[1] www.ftdichip.com/Products/ICs/FT2232H.htm
[2] www.elektor.de/110534
[3] www.ftdichip.com/Support/Utilities/MProg3.5.zip
[4] www.ftdichip.com/Drivers/CDM/CDM20814_Setup.exe

Audio-DSP-Kurs
Teil 5: Aufbau der DSP-Kursprogramme

Im vorliegenden Artikel beschreiben wir die Strukturen der für die Anwendungen im DSP-Kurs entwickelten DSP-Programme. Wir haben uns für eine Aufteilung in ein für alle Anwendungsprogramme identisches Rahmenprogramm und individuelle Programme für die Audiosignalverarbeitung in der Audioloop entschieden. Damit möchten wir den Einstieg in die DSP-Programmierung und in die Verwendung der Kurs-Programme erleichtern. Der Artikel wird mit nützlichen Hinweisen und Tipps zur Verwendung des Assemblers und zur Programmierung der DSPs beendet.

Von Alexander Potchinkov (D)

Wir teilen die Software in zwei Teile auf. Der erste Teil, den wir als *Rahmenprogramm* bezeichnen, ist für alle Anwendungen gleich. Der zweite Teil ist die *Audioloop* mit der Audiosignalverarbeitung, die für die Kursprojekte und mögliche weitere Anwendungen der Leser individuell ausgeführt wird. Beide Begriffe haben wir bereits im zweiten Artikel dieses Kurses eingeführt. Jetzt werden wir sie auf das DSP-Board und die Anwendungen in diesem Kurs übertragen und spezialisieren.

Aufgaben des Rahmenprogramms

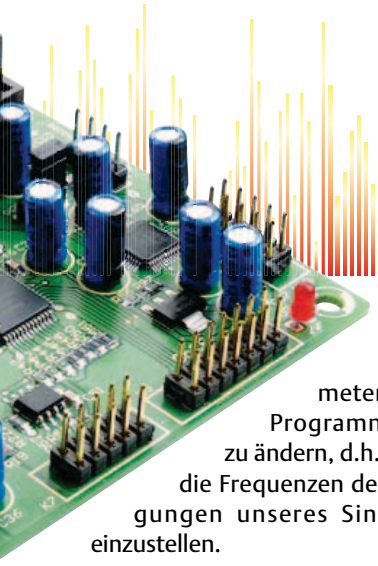
Mit dem Rahmenprogramm wird zum einen der DSP selbst eingestellt: Wir legen Prozessortakt und Interruptsystem fest, richten einen Software-Stack ein und Weiteres mehr. Zum anderen wird der DSP mit den Peripheriekomponenten des DSP-Boards verbunden, um den Audio- und Steuerdatenaustausch mit ihnen zu ermöglichen. Verbinden mit den Peripheriekomponenten bedeutet, die DSP-Register zu beschreiben, mit denen die Eigenschaften der DSP-Schnittstellen festgelegt werden. Man kann sagen, dass mit dem Rahmenprogramm die Funktionsweise der Hardware eingestellt und der Betrieb der Hardware ermöglicht wird. Dies betrifft in unserem Fall fünf Peripheriekomponenten, nämlich ADC, DAC, SRC, SEEPROM und SPI-Port, die wir uns als „Planeten um die DSP-Sonne“ vorstellen können. **Bild 1** zeigt dieses „Sonnensystem“. Die durchgezogenen Linien stellen die Datenübertragung und die unterbrochenen Linien die Übertragung von Steuersignalen dar, die einerseits zum Programmieren und andererseits zur Übermittlung von Informationen über Betriebszustände der Peripheriekomponenten dienen.

- Für den **ADC** werden die Taktsignale zur Verfügung gestellt, mit denen die Abtastfrequenz und die Zeitsteuerung der Datenübertragung festgelegt werden. Der ADC ist in der Lage, das vom DSP vorgegebene Verhältnis von Mastertaktfrequenz und Abtastfrequenz selbst zu erkennen und muss daher auch nicht programmiert werden. In unserem Fall hat das Verhältnis den Wert 512.

- Für den **DAC** werden ebenfalls die Taktsignale zur Verfügung gestellt. Auch der DAC ist wie der ADC in der Lage, die vom DSP vorgegebenen Taktfrequenzverhältnisse zu erkennen und korrekt umzusetzen.

- Der **SRC** kann nicht so „selbständig“ wie ADC und DAC betrieben werden, sondern er muss programmiert werden. Mit der Bytefolge aus dem dritten Teil der Artikelserie werden die I²S- und Digital-Audio-Schnittstellen des SRCs konfiguriert. Die Programmierung erfolgt über den DSP via SPI und ist Teil des Rahmenprogramms. Hierfür muss das SPI so eingestellt werden, dass es dem SPI-Übertragungsprotokoll des SRCs entspricht. Anschließend wird der SRC vom DSP zurückgesetzt und es werden 54 Bytes zum SRC übertragen.

Das **SEEPROM** kann als allgemeiner nichtflüchtiger Speicher genutzt werden. Falls ein Betrieb des DSP-Boards mit einer fest vereinbarten Anwendung ohne Benutzung des Debuggers gewünscht ist, kann man den DSP aus dem SEEPROM heraus booten. Der DSP enthält hierfür ein Bootloader-Programm in einem ROM-Bereich.
Mit dem **SPI-Port** kann ein Datenaustausch mit Komponenten außerhalb des DSP-Boards betrieben werden. Für den digitalen Aussteuerungsmesser in unserem Kurs nutzen wir den SPI-Port zur Ansteuerung der LED-Balkenanzeige. Dann können wir auch am SPI-Port z.B. einen Mikrocontroller mit Tastenfeld, Digitalpotentiometer (Drehencoder) und LCD anschließen, um die Para-



meter unserer DSP-Programme im Betrieb zu ändern, d.h. beispielsweise die Frequenzen der Sinusschwingungen unseres Sinusgenerators einzustellen.

Aufbau des Rahmenprogramms

Das Rahmenprogramm besteht aus vier Teilen:

1. Allgemeine Deklarationen

- Festlegung der Speicherbelegung durch die Programmvariablen und Parameter. Hierfür haben wir feste Speicherbereiche vorgesehen.
- Wertzuweisungen an vom Programm benötigte Konstanten
- Einträge in der Interruptvektortabelle, die in der Datei `ivt.asm` abgelegt sind.

2. DSP-Einstellungen

Die Einstellungen am DSP (die Konfigurationen des Prozessorkerns und der Peripherieschnittstellen) erfolgen über das Schreiben von 24-bit-Registern, die über Adressen aus dem am oberen Adressende liegenden Adressraum $FFFF80 bis $FFFFFF angesprochen werden. Dieser Adressraum schließt 128 Adressen ein, von denen wir bis zu 18 nutzen, und wird vom Prozessorhersteller als *Internal I/O Memory Map* bezeichnet. Für unseren DSP56374 im 52-Pin-Gehäuse wird nur das X-RAM benutzt. Das Schreiben der Register erfolgt mit dem speziellen Befehl `movep` (Move Peripheral Data), der es erlaubt, direkt in das ausgewählte Register zu schreiben, ohne den Umweg über ein Prozessorregister wie beim „gewöhnlichen“ `move`-Befehl nehmen zu müssen. Ein Beispiel ist der Befehl `movep #$D17D00,x:RCR`, mit dem das RCR (Receive Clock Register) gesetzt wird. Dieses Register legt Eigenschaften des Empfangsteils der Audioschnittstellen fest. `RCR` ist eine sinnvolle Abkürzung (ganz wie die Mnemonics der Assemblersprach-Befehle) hinter der sich die Adresse $FFFFBF verbirgt, die man sich kaum merken kann. Aus diesem Grund verwenden wir in jedem Programm die Hilfsdatei `mioequ.asm`, in der unter anderem alle I/O-Register-Adressen mit sinnvoll abkürzenden Bezeichnern verknüpft werden, was dem Programmierer die Arbeit erheblich erleichtert. Diese Datei wird mit der Assembler-Include-Anweisung eingebunden.

- Programmbeginn beim Programmzählerstand $000000, der nach dem DSP-Reset eingenommen wird. Die erste Anweisung ist ein Sprung an die Programmadresse $100, die hinter der Interruptvektortabelle und weiteren DSP-Interna liegt. Jetzt beginnt die eigentliche Programmabarbeitung.
- Blockieren aller Interrupts für die nun folgenden Einstellungen am Prozessor. So können unerwartete oder blockierende „Unfälle“ vermieden werden, während der Prozessor über seine Kern- und Peripherieregister initialisiert wird.
- Rücksetzen des Hardware-Stackpointers sp und Einrichten des Software-Stacks im X-RAM ab Adresse $40 mit Adressen in aufsteigender Richtung.

Die Einstellungen am DSP und seiner Peripherieschnittstellen sind recht aufwendig, da der DSP sehr flexibel und universell eingerichtet ist. Wir wollen nicht verschweigen, dass dieses Einstellen eines DSPs nicht ohne ausgiebiges Studium des Manuals möglich ist und auch eine wahre Anfängerhürde darstellt. Glücklicherweise ist der Prozessor „recht logisch“ aufgebaut und die Freescale-Manuals sind gut geschrieben. Eine ausführliche Schritt-für-Schritt-Beschreibung würde den hier zur Verfügung stehenden Rahmen allerdings bei weitem sprengen. Daher geben wir im Folgenden nur die wichtigsten Hinweise:

- **PLL-Einstellungen für den Prozessortakt**. Der Prozessortakt wird auf 147,456 MHz eingestellt, was dem sechsfachen der Audio-Masterfrequenz und dem 3072-fachen der Abtastfrequenz von 48 kHz entspricht. Bei der Festlegung der Teiler- und Vervielfacherfaktoren ist der zulässige VCO-Frequenzbereich von 300 MHz bis 600 MHz zu beachten (wir betreiben den VCO bei einer Frequenz von 589,824 MHz). Wir finden im Handbuch in Tabellen geeignete Hinweise für die gebräuchlichen Audiofrequenzen. Bei davon abweichenden Frequenzen müssen wir die notwendigen Berechnungen selbst vornehmen. Wir gehen davon aus, dass das Board mit einem 24,576-MHz-Quarzoszillator bestückt ist. Die PLL ist dann mit `PCTL=#$01E006` auf einen Takt-Multiplikationsfaktor von 6 einzustellen.
- Mit der Anweisungsfolge `movec #0,sp`, `move #$40,r6` und `move #-1,m6` werden der **Hardware-Stack-Pointer** zurückgesetzt und der **Software-Stack** eingerichtet.

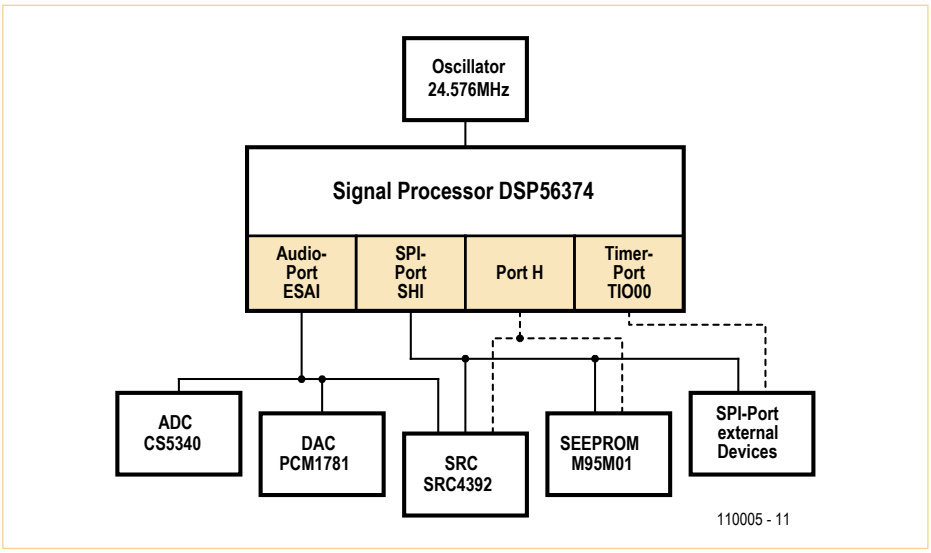


Bild 1. DSP-Sonnensystem.

- Mit `IPRP=#$000003` wird das **Interrupt-System** für die Interrupts der Audioschnittstellen geöffnet und auf den Prioritätenlevel 2 priorisiert. Die Core-Interrupts nutzen wir in diesem Kurs nicht.

- **Audio-Schnittstellen**: Einzustellen sind der Master-Betrieb mit der Mastertaktfrequenz von 24,576 MHz, der Netzwerkbetrieb mit zwei Kanälen und 24-bit linksbündigen Daten im 32-bit-Rahmen, I²S, Interrupts für Empfangen und Senden, Ausnahmebehandlung und Last-Slot (rechter Kanal). Die Empfänger- und Senderregister werden mit gleichen Inhalten geschrieben `RCCR=TCCR=$FDD302` und `RCR=TCR=$D17D00`.
Mit `RSMA=RSMB=TSMA=TSMB=#$00FFFF` werden im Netzwerkbetrieb die Kanäle (es können bis zu jeweils 32 sein) einzeln für Sender und Empfänger freigegeben oder gesperrt. Bei der hier gewählten Einstellung werden alle der möglichen Kanäle freigegeben, von denen wir nur zwei nutzen. Mit `PCRC=PRRC=#$000FFF` wird der Port C als Audioport konfiguriert. Es ist möglich, auch einzelne Pins im Port C als GPIO-Pins zu konfigurieren und bei unserem DSP-Port am Audiostecker im Bedarfsfall auch zu nutzen.

- **SHI-Schnittstelle**: Einzustellen sind SPI-Modus, der Master-Betrieb mit der Mastertaktfrequenz von 0,9216 MHz, keine Interrupts, `cpol=cpha=0`, narrow spike filter, FIFO off.
Programmierung des SRC: 8-bit-Daten, Einstellungen `HCKR=#$002048` und `HCSR=#$000040`
Ansteuerung der LED-Balkenanzeige: 16-bit-Daten, Einstellungen `HCKR=#$002048` und `HCSR=#$000044`
- Konfiguration des **Port H**: Der Port H wird für den SRC und das SEEPROM sowie für die Festlegung des Boot-Modus verwendet.
PH4: GPIO-Input, Lock-Signal vom SRC4392
PH3: GPIO-Output, Reset des SRC4392
PH2: Betrieb als MODC-PIN, Bootmodus des DSPs
PH1: GPIO-Output, ChipSelect des SRCs SRC4392
PH0: GPIO-Output, ChipSelect des SEEPROMs M95M01.
Die Konfiguration erfolgt mit `PCRH=#$000014` und `PRRH=#$00000F`.

3. Programmteile vor Ausführung der Audioloop

- Interrupts freischalten, insbesondere die Audiointerrupts, damit die Buffer gelesen und geschrieben werden und die flaggesteuerte Synchronisation auf empfangene Audiodaten ermöglicht wird.
- Programmierung des SRC über das SHI im SPI-Modus.

4. ISR

- Zum Rahmenprogramm gehören am Ende der Programmdatei die Programmteile für die ISR, die in der Datei `esai4R2T.asm` abgelegt sind.

Audioloop

Die Audioloop enthält die digitale Audiosignalverarbeitung und wird in das Rahmenprogramm eingebettet. Wir verfolgen das Konzept, die Blockstruktur der Signalverarbeitung durch Unterprogramme widerzuspiegeln, indem ein Block eins zu eins durch ein Unterprogramm aufgebaut wird. Die Vorstellung von Signalverarbeitung in Blöcken ist auch dem mit Analogtechnik vertrauten Leser wohlbekannt.

Unsere Unterprogramme haben konsequenterweise *Ein-* und *Ausgangssignale*. Im folgenden Beispiel sind es die vier Signale `SignalInL/R` und `SignalOutL/R`. Davon weicht zum Beispiel ein Signalgenerator ab. In diesem Fall hat das Unterprogramm nur Ausgangssignale. Signale werden in unseren Projekten in einfacher Genauigkeit mit 24 bit repräsentiert und belegen jeweils eine Speicherzelle, die in einem festgelegten Signal-Speicherbereich im DSP-RAM liegt.

Des Weiteren werden Unterprogramme mit Parametern in ihrer Arbeitsweise festgelegt. Unter Parametern fassen wir die Einstellparameter der Signalverarbeitung und die Parameter der DSP-Programme zusammen. Solche Parameter können zum Beispiel Zeitkonstanten sein. **Bild 2** zeigt eine mögliche Ablauffolge von Unterprogrammen. Vier Unterprogramme, `Subroutine A` bis `Subroutine D`, repräsentieren vier Signalverarbeitungsblöcke. Die rechteckigen Kästchen in der Mitte der Abbildung stellen die Unterprogramme dar, die über Signalpfade miteinander in Verbindung stehen. Die Kästchen mit den gerundeten Kanten symbolisieren die Parameter der Unterprogramme. Nicht jedes Unterprogramm benötigt Parameter, wie es in der Abbildung beim Unterprogramm C der Fall ist. Die spitz zulaufenden Kästchen enthalten die Signale, die zwischen den Unterprogrammen vorliegen und symbolisieren „Oszilloskope“ zu ihrer Beobachtung. Diese Signale haben eigene Speicherplätze im DSP-Programm und können daher auch zu jedem Zeitpunkt beobachtet werden.

Wir haben so die Block-Signalverarbeitung nachgebildet und eine Möglichkeit geschaffen, mit der Beobachtung der Signale die Funktionsweise und das Funktionieren der Signalverarbeitung zu erkennen. Wir können beispielsweise einzelne Signale am Ende der Audioloop auf die Audioausgabe-Buffer legen und sie mit einem Digital-Audio-Oszilloskop beobachten. Das hilft uns in erheblichem Maße, Fehler in der Signalverarbeitung aufzuspüren. Der Autor verwendet hierzu eine preiswerte Standard-USB-Soundkarte, wie man sie für deutlich weniger als 100 Euro kaufen kann, und einen Waveeditor. Es sind kommerzielle Waveeditoren erhältlich. Es können aber kostenfreie Waveeditoren und Audioanalyser über das Internet beschafft werden. Eine solche Software erlaubt das Betrachten eines Signals (oder auch mehrerer) im Zeitbereich und des Spektrums im Frequenzbereich.

Wir verwenden diese Software wie ein leistungsfähiges Oszilloskop. Am Spektrum kann man zum Beispiel die Klirranteile eines Sinusgenerators erkennen. Schließlich lassen sich mit einem Wave-Editor Sounddateien erzeugen und vielfältig bearbeiten. Sounddateien kann man auch in Numeriksoftware wie Matlab laden und nach allen Regeln der Kunst analysieren. Ein Beispiel hierfür ist der Test eines Dynamikprozessors mit Burst-Signalen einstellbarer Dauer, Frequenz und Amplitude. Da auf dem DSP-Board ein DAC vorhanden ist, können wir die Signale auch mit einem Analogoszilloskop beobachten, was aber eher dem Verschaffen eines Überblicks und weniger einer möglichen präzisen Analyse dient.

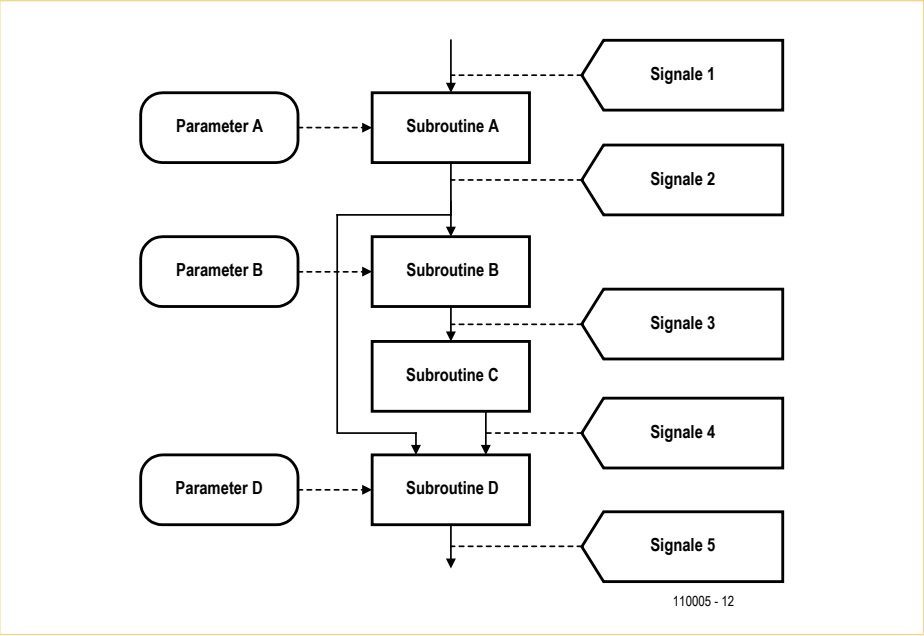


Bild 2. Beispielunterprogramme in einer Audioloop.

Der ADC ist an SDI1(SDO4) und der SRC an SDI2(SDO3) angeschlossen. Im Empfangsbuffer, der über vier Speicherstellen verfügt, lautet die Aufteilung

```
x:RxBuffBase
                  ADC, linker Kanal
x:RxBuffBase+1
               SRC.RX, linker Kanal
x:RxBuffBase+2
                 ADC, rechter Kanal
x:RxBuffBase+3
             SRC.RX, rechter Kanal.
```

Der DAC und der AES3-Coder mit Sender im SRC sind an SDO0 angeschlossen. Im Sendebuffer, der über zwei Speicherstellen verfügt, lautet die Aufteilung

```
x:TxBuffBase
       DAC und SRC.TX, linker Kanal
x:TxBuffBase+1
     DAC und SRC.TX, rechter Kanal.
```

Am Anfang und am Ende enthält die Audioloop noch zwei Programmstückchen zur Audiotaktsynchronisation und zum Lesen der Audioempfangsbuffer sowie zum Schreiben der Audiosendebuffer.

```
Audioloop
   jclr  #RightRx,x:LRFlag,*
   bclr  #RightRx,x:LRFlag
   move    x:RxBuffBase,a
                ; ADC CS5340, left
   move    x:RxBuffBase+2,b
              ; ADC CS5340, right
   brset #Lock_SRC4392,x:PDRH,NoSRC
     ; use ADC if SRC does not lock
     move    x:RxBuffBase+1,a
             ; SRC.RX SRC4392, left
     move    x:RxBuffBase+3,b
           ; SRC.RX SRC4392, right
NoSRC move    a,y:InL
   move    b,y:InR
```

Mit den ersten beiden Zeilen wird die Synchronisation durchgeführt, wie es im Teil-2-Artikel beschrieben wurde. Mit der dritten und der vierten Zeile werden die ADC-Signale in die Akkumulatorregister a und b geschrieben. In der fünften Zeile wird das Lock-Bit des SRCs abgefragt. Wenn das Lock-Bit ein gültiges Audiosignal anzeigt, werden die SRC-Signale in die Akkumulatorregister a und b geschrieben. Im Falle eines ungültigen Audiosignals wird dieses Überschreiben übersprungen. In der achten und neunten Zeile werden die Signale aus den Akkumulatorregistern in die Signalspeicher `InL` und `InR` geschrieben, um vom ersten Signalverarbeitungsunterprogramm gelesen und verarbeitet zu werden. Mit dieser Technik stellen wir sicher, dass stets ein Audiosignal gelesen wird. Wenn man ein Digitalsignal an das Board anschließt, wird es bevorzugt verarbeitet. Wenn kein gültiges Digitalsignal vorhanden ist, wird das ADC-Signal verarbeitet.

Das Ende der Audioloop wird vom Schreiben der Sendebuffer mit den Inhalten der Akkumulatorregister a und b, die zuvor mit den Signalen `OutL` und `OutR` geladen wurden, und dem Rücksprung an den Beginn der Audioloop gebildet.

```
   move    y:OutL,a
   move    y:OutR,b
   move    a,x:TxBuffBase
          ; -> DAC and SRC.TX, left
   move    b,x:TxBuffBase+1
         ; -> DAC and SRC.TX, right
   jmp     AudioLoop
```

Speicherbelegungen

In **Tabelle 1** ist die Speicherbelegung für die Kurssoftware angegeben. Wir haben weniger darauf geachtet, den Speicher lückenlos zu füllen, sondern vielmehr darauf, mit festgelegten Bereichen die Vergleichbarkeit der Kursprogramme zu verbessern. Nicht in jedem Programm werden auch alle der bezeichneten Speicherbereiche belegt.

Adresszeiger, Software-Stack und Sample-Counter

Zwei der acht Adresszeiger `R0` bis `R7` werden im Rahmenprogramm für „übergeordnete Aufgaben“ verwendet und dürfen daher nicht vom Anwendungs-DSP-Programm verwendet werden. Der Zeiger `R6` wird für den *Software-Stack* im X-RAM verwendet. Dieser Software-Stack ergänzt den Hardware-Stack und nutzt den Adressraum über $40 im X-RAM. Für den Software-Stack stehen in unseren Programmen 32 Speicherstellen im Bereich $40...$5F zur Verfügung.

In unserem Rahmenprogramm wird der Software-Stack von den Audio-ISR benutzt, um den Zeiger `R0` mit Attribut `M0` zu sichern und nach Abarbeiten der ISR wieder herzustellen. Dabei muss man stets daran denken, dass ISR nicht Register manipulieren dürfen, die im Programm verwendet werden. Dieser Software-Stack kann auch von Benutzerprogrammen für Prozessorregister, also Register im Datenpfad und vor allem Register in der AGU, genutzt werden. Der Zeiger `R7` dient als Modulo-Abtastzeitintervallzähler. Wir verwenden eine Modulo-192-Zählung. Dieser Zähler, wir bezeichnen ihn als *Sample-Counter*, wird zweifach benutzt. Zum einen, um in den Testprogrammen die beiden Länge-192-Sinussignale in

```
Sinussignal, f=1kHz,
                    x:$800..x:8BF
Sinussignal, f=2kHz,
                    y:$800..y:8BF
```

auszulesen, die dann über den DAC oder den Digitalausgang ausgegeben werden können. Die zweite Verwendung ist das Auslösen des zyklischen Schreibens der Daten zum LED-Display beim digitalen Aussteuerungsmesser, unserer zweiten DSP-Anwendung. Der Schreibtakt beträgt dann 48 kHz/192 = 250 Hz, was heißt, dass das Display alle 4 ms mit Daten beschrieben wird. Den Zeiger `R7` könnte man in einem Anwenderprogramm anderweitig nutzen, wenn er mit dem Stack gesichert und nach seiner Nutzung auch wieder aus dem Stack wiederhergestellt wird.

Unterprogramme

Ein Unterprogramm, auch als Subroutine bezeichnet, hat die Struktur

```
NameDesUnterprogramms
   move  y:SignalInL,x0
   move  y:SignalInR,y0
   <Signalverarbeitung, ggf.
abhängig von Parametern>
   move  x0, y:SignalOutL
   move  y0, y:SignalOutR
   rts
```

Das Unterprogramm wird mit dem Befehl `jsr NameDesUnterprogramms` aufgerufen. Der Befehl `jsr` (jump to subroutine) veranlasst den DSP, die Inhalte von Prozessorzustandsregistern auf dem Hardware-Stack abzulegen und die Unterprogrammbefehle abzuarbeiten.

Das Unterprogramm wird mit dem Befehl `rts` (return from subroutine) verlassen, das heißt, der DSP stellt die Prozessorzustandsregister wieder her und kehrt zum aufrufenden Programm zurück. Eine Parameter-Übergabe an Unterprogramme kann über den Software-Stack erfolgen

Makros

Ein DSP-Makro ist ein Textmakro, das der Assembler an der Stelle in das Programm einsetzt, an der das Makro aufgerufen wird. Ein Makro hat die Struktur

```
NameDesMakros  macro  param1
param2 .. paramN

   endm
```

Der Aufruf des Makros erfolgt mit der Angabe des Makronamens und der möglichen Parameter. Mit solchen Makros lassen sich Programme übersichtlicher schreiben, zumal die Makros auch die Übergabe von Parametern ermöglichen. Allerdings wird der Programmumfang bei mehrfacher Benutzung des Makros in einem Programm nicht verringert. Im Gegensatz dazu stehen Unterprogramme, die helfen können, sowohl übersichtlichen Code zu erstellen als auch den Codeumfang zu beschränken. Dafür benötigen sie aber Prozessortakte mit der Bedienung des Stacks beim Aufrufen und beim Verlassen.

Initialisierungen der Signalverarbeitung

Direkt vor Beginn der Audioloop werden zwei Unterprogramme ausgeführt:
- Das Unterprogramm `ZeroState`, mit dem die Zustandsspeicher der Signalverarbeitung auf Null gesetzt werden.
- Das Unterprogramm `SetDefaultParams`, mit dem die Defaultwerte der Signalverarbeitungsparameter gesetzt werden. In diesem Unterprogramm kann man durch Ändern der Defaultwerte die Signalverarbeitung an eigene Vorgaben anpassen. Dies kann auch über das SPI mit einer Eingabeeinheit erfolgen.

Laden und Ausführen der Programme

Wenn ein DSP-Programm fertiggestellt ist, wird es - vorausgesetzt, dass es fehlerfrei ist - assembliert. Das kann im cmd-Fenster mit dem Aufruf

```
asm56300 -a -b -l myprogram.asm
```

erfolgen. Mit der Option –l wird der Assembler dazu angehalten, ein List-File anzulegen, was bei der Fehlersuche oft sehr nützlich ist. Das assemblierte Programm `myprogram.cld` kann nun in den Debugger geladen werden, an den der DSP angeschlossen ist. Vor dort aus lässt es sich mit einem Adapter auf den DSP schreiben und seine Ausführung veranlassen.

Nützliche Hinweise

Zur Verwendung des Assemblers:

- Zeichen an der ersten Textposition interpretiert der Assembler als Label oder Marke. Ein Label beginnt mit einem Buchstaben und darf kein reserviertes Schlüsselwort wie `move` oder `a0` sein.
- Der DSP-Assembler ist ein Zwei-Pass-Assembler. Ein Zwei-Pass-Assembler legt im ersten Lauf eine Symbol-Tabelle für alle verwendeten und dereferenzierten Label an. Im zweiten Lauf übersetzt er die Befehle und kann wegen der zuvor angelegten Symboltabelle gleichzeitig assemblieren und Verweise auf relativ zur Position des Assemblierens später erklärte Label verarbeiten, die sonst zum Zeitpunkt des Assemblierens einer Stelle noch unbekannt wären.
- Nützlich für die Programmierung von Makros ist das Label, das mit dem Unterstrich _ beginnt, da es sich dann um ein lokales Label handelt.
- Im Gegensatz zu den Schlüsselwörtern werden bei Labels Groß- und Kleinschreibung unterschieden.
- DSP-Befehle werden ab der zweiten Textposition geschrieben.
- Die Datei `mioequ.asm` wird mit `include` in ein DSP-Programm eingebunden, da sie die zahlreichen Kurznamen des DSP-Manuals für die Schnittstellenregister zur Verfügung stellt.
- Die niedrigste Programmstartadresse ist p:$100, denn an den niedrigeren Adressen liegen die Interruptvektortabelle und reservierte Bereiche.
- Die org-Anweisung verlangt die Angabe eines Speicherbereichs X, Y, P oder L.
- Wegen der Befehlspipeline, die, wenn sie nicht zum Beispiel durch Programmverzweigungen oder Long-Interrupts unterbrochen wird, die Befehlsausführungszeit erheblich reduziert, muss der Assembler ab und an `nop`-Befehle in den Object-Code einfügen. Hierauf weist er mit Warnhinweisen hin, die dem Programmierer Anlass geben sollten, seinen Code auf Effizienz hin zu überdenken und gegebenenfalls Befehle zu vertauschen, damit statt eines `nop`-Befehls etwas „Sinnvolles“ ausgeführt wird.
- Im Prozessorhandbuch befinden sich am Ende einige so genannte „programming sheets“, die das Programmieren der Prozessorregister erheblich vereinfachen. Wir

Tabelle 1. Speicherbelegung für die Kurssoftware.			
Belegung X-RAM	**Belegung Y-RAM**	**Bereich**	**Belegung P-RAM**
Audio RX-Buffer	Signale	$00..$0F	Interrupt-Vektortabelle und reservierte Bereiche
Audio TX-Buffer	Signale	$10..$1F	
Audio Flags und Zeiger	Zeiger, Koeffizienten, Zustandsspeicher	$20..$2F	
Programmparameter		$30..$3F	
Software-Stack	Zur freien Verwendung	$40..$4F	
Software-Stack	Zur freien Verwendung	$50..$5F	
Zur freien Verwendung		$60..$9F	
Hilfsvariablen		$A0..$BF	
Bootsequenz SRC	Zur freien Verwendung	$C0..$CF	
Zur freien Verwendung		$D0..$FF	
Filter- und Polynomkoeffizienten		$100..$5FF	Anwenderprogramme
Ringspeicher, Links	Ringspeicher, Rechts	$600..$7FF	
Sinussignal 1 kHz	Sinussignal 2 kHz	$800..$8BF	
Zur freien Verwendung		$8C0..$17FF	

raten jedem Programmierer zur Verwendung dieser Blätter, die sich jederzeit aus dem PDF-Manual heraus ausdrucken lassen und für die Programmerstellungsphase benutzt werden sollten, um dann schließlich der Dokumentation hinzugefügt zu werden.

Zur Programmierung des DSPs:

- Direkte Wertzuweisungen (immediate moves) zu Speicherzellen wie `move #$123456,x:$000100` sind nicht möglich und müssen über den Umweg eines Registers wie zum Beispiel `move #$123456,x0 move x0,x:$000100` ausgeführt werden. An die Peripherieregister im höchsten Adressraum können direkte Wertzuweisungen vorgenommen werden wie `movep #$123456,x:$FFFFE0`.
- Auch bei einem Datentransfer von Speicherzelle zu Speicherzelle müssen Prozessorregister als „Zwischenlager“ wie `move x:$000010,x0 move x0,y:$000010` verwendet werden. Bei Peripherieregistern sind solche Datentransfers aber möglich.
- Die Anweisung `move #$F,x0` führt nicht zum erwarteten Ergebnis `x0=$00000F`, sondern zu `x0=$0F0000`. Mit `move #>$F,x0` erreicht man das rechtsbündige Ergebnis. Der DSP ist ein Fractional-Prozessor, das heißt, die Zahlen werden linksbündig abgelegt.
- Verschachtelte Do-Schleifen müssen auf unterschiedliche Labels verweisen. Im Weiteren müssen zwischen den Labels wenigstens `nop`-Befehle eingefügt werden, falls nicht andere Befehle eingetragen sind.
- Das P-RAM sollte nicht mit Daten belegt werden, da ein Zugriff zusätzliche Prozessorzyklen kostet.
- Wenn Integerzahlen, beispielsweise für Adressberechnungen, miteinander multipliziert werden müssen, ist es erforderlich, das Ergebnis mit einem Rechtsshift `asr` zu halbieren, da der DSP einen Multiplizierer für linksbündige Fraktionals verwendet und Integerzahlen rechtsbündige Zahlen sind.
- Bei den Drei-Adressbefehlen `mac` und `mpy` können nicht alle 16 möglichen Kombinationen der vier Operandenregister verwendet werden, da zur Codierung der zu verwendenden Operandenregister nur drei und nicht vier Bits vorgesehen sind. So sind zwar `mpy x0,x0,a` oder `mac x0,y1,b`, nicht aber `mpy x1,x1,a` oder `mpy y1,x0,b` möglich.
- Fast Interrupts sind nicht mit dem `rti`-Befehl abzuschließen.
- Den Zeiger `R6` haben wir für den Software-Stack reserviert, der im X-Ram die Basisadresse X:$40 hat. Den Software-Stack benötigen wir im Besonderen dann, wenn in Unterprogrammen AGU-Register manipuliert werden. Ein „beliebter“ Fehler ist es, in einem Filter-Unterprogramm eine Modulo-Adressierung zu erzwingen, die außerhalb des Filter-Unterprogramms unerwünscht ist und im Unterprogramm noch nicht zurückgenommen wurde.
- Zu Programmbeginn sollte der Stackpointer des Hardwarestacks `sp` mit Null initialisiert werden.
- Bei besonders zeitkritischen Anwendungen sollte man die Benutzung der Befehls-Extension vermeiden. In dieser Extension, dem möglichen zweiten Wort eines Befehls, stehen zum Beispiel Adressen oder Zahlwerte. Der Befehl `mpyi #0.3,x1,a` benötigt die Extension für die Fraktionaldarstellung der Dezimalzahl 0.3 und belegt daher zwei Befehlsworte. Der Befehl `mpy x0,x1,a` hingegen benötigt nur ein Befehlswort. Man sollte daher zum Beispiel Konstanten zu Programmbeginn im DSP-RAM ablegen und bei Bedarf (möglicherweise mit einer Parallelausführung) rechtzeitig vor der Benutzung in ein Prozessorregister, hier das Operandenregister `x0`, schreiben.
- Wenn man ein DSP-Programm betrachtet, sieht man, dass `move`-Befehle sehr häufig verwendet werden, was seine Ursache unter anderem darin hat, dass der DSP ein Registerprozessor ist. Alleine deshalb sollte man die Fähigkeit des DSPs zu parallelen Daten-Moves ausnutzen. Oft heißt dies, dass man Register (schon lange) vor ihrer Nutzung schreibt, wenn es parallel zu einer arithmetischen Operation möglich ist. Leider wird der Programmcode dadurch sehr unübersichtlich, da die zusammengehörenden Codesegmente räumlich verteilt werden.
- Nach einigen Jahren des DSP-Programmierens haben wir gelernt, dass die häufigsten Fehler durch unbeabsichtigt mehrfach genutzte Speicherstellen auftreten. Wir empfehlen daher, für jedes DSP-Programm eine detaillierte und vollständige Liste der Speicherbelegung anzulegen. Auch die von uns gewählte Technik, Variablengruppen feste Speicherbereiche zuzuweisen, ist vielleicht hinsichtlich der vollständigen Nutzung des Speichers nicht effizient, dafür ist sie aber weniger fehleranfällig.
- Für spätere Programme ist es nützlich, zwei Routinen zu schreiben, mit denen die 10 Datenpfad-Register `x0, y0, x1, y1, a0, b0, a1, b1, a2` und `b2` auf den

Tabelle 2. Unterschiede zwischen Zweierkomplement-Integer-Zahlen und Fractionals.

Integer	Dezimal	Integer	Dezimal	Fractional	Dezimal	Fractional	Dezimal
0000	0	1000	-8	0,000	0	1,000	-1,0
0001	1	1001	-7	0,001	0,125	1,001	-0,875
0010	2	1010	-6	0,010	0,25	1,010	-0,75
0011	3	1011	-5	0,011	0,375	1,011	-0,625
0100	4	1100	-4	0,100	0,5	1,100	-0,5
0101	5	1101	-3	0,101	0,625	1,101	-0,375
0110	6	1110	-2	0,110	0,75	1,110	-0,25
0111	7	1111	-1	0,111	0,875	1,111	-0,125

Software-Stack geschrieben und zurückgelesen werden können. Man kann diese Stackbenutzung für Unterprogramme verwenden, um ungewünschte Seiteneffekte mit der Registernutzung im aufrufenden Programm zu vermeiden.

Zahlenformat der Signale im DSP

Für die Signale im DSP verwendet man ein spezielles Zahlenformat, bei dem das Binärkomma direkt hinter dem ersten Bit, dem *Vorzeichenbit* angeordnet ist. Es ändert sich so nichts weiter als die Stellenwertigkeit gegenüber den Integerzahlen, bei denen man kein Komma verwendet oder bei denen man sich ein Komma rechts von den Zahlen denken kann. Dieses Zahlenformat bezeichnet man als *Zweierkomplement-Fractionals* oder kurz *Fractionals*. Die Unterschiede zwischen Zweierkomplement-Integer-Zahlen und Fractionals lassen sich in einfacher Weise an 4-bit-Zahlen aufzeigen, von denen es 16 unterschiedliche gibt, 7 positive Zahlen, 8 negative Zahlen und die Null, siehe **Tabelle 2**.

In der Tabelle wurde zur besseren Lesbarkeit das Vorzeichenbit markiert und bei den Fractionals ein Komma notiert. Die Zahlen unterscheiden sich mit ihren Stellenwertigkeiten, die bei den 4-bit-Integerzahlen rechts vom führenden Vorzeichenbit von links nach rechts die Werte 4, 2 und 1 aufweisen und bei den Fractionals hinter dem Komma ½, ¼, $^1/_8$. Beim Integer-Format werden die Zahlen rechtsbündig und beim Fractional-Format linksbündig eingetragen. Darunter versteht man, dass bei Integerzahlen links von der Zahl das Hinzufügen von Nullen bei positiven Zahlen oder das Hinzufügen von Einsen bei negativen Zahlen den Wert der Zahlen nicht ändert. Bei Fractionals trifft dies für das Hinzufügen von Nullen rechts von einer Zahl zu, sowohl bei positiven als auch bei negativen Zahlen. Dem aufmerksamen Leser wird nicht entgangen sein, dass die Bereichsgrenzen für positive und negative Zahlen unterschiedlich sind. Das verursacht an manchen Stellen Probleme, denn wir sehen darin die Grenzen unserer Aussteuerbereiche. Beim 24-bit-DSP beträgt der Werteunterschied $2^{-23} = 1{,}1921 \cdot 10^{-7}$, was in manchen Fällen vernachlässigt werden kann.

Nun kommen wir zum Grund der Verwendung von Fractionals. Wir verwenden einen abgespeckten DSP, der in einfacher Genauigkeit mit 4-bit-Zahlen und in doppelter Genauigkeit mit 8-bit-Zahlen arbeitet, die aber wie beim 24-bit-DSP zu interpretieren sind und nur eine viel geringere Auflösung (das ist der Abstand zweier direkt aufeinanderfolgender Zahlen) aufweisen. Dieser DSP muss nun eine *Requantisierung* durchführen, was eine in der digitalen Signalverarbeitung sehr häufig auszuführende Operation ist. Im Beispiel muss eine 8-bit-Zahl zu einer 4-bit-Zahl requantisiert werden. Die 8-bit-Zahl lautet:

0,100 0100 Zahlenwert: ½ + $^1/_{32}$ = 0,53125

Die Requantisierung erfolgt in diesem Beispiel durch *Abschneiden* der hinteren vier Stellen, womit man die Zahl

0,100 Zahlenwert: ½ = 0,5

erhält. Das sieht vielleicht noch nicht sehr eindrucksvoll aus, der Leser sollte sich aber überlegen, wie ein vergleichbarer Vorgang bei Integerzahlen auszuführen ist.

Zurück zum DSP: Der DSP rechnet in einfacher Genauigkeit mit 24 bit und in doppelter Genauigkeit mit 48 bit. Wir wissen, dass bei Digital-Audio die Bezugswortbreite 24 bit ist, was beim DSP der einfachen Genauigkeit entspricht. Aber was passiert, wenn wir zwei Signalwerte miteinander multiplizieren oder, wenn wir in einem einfachen Abschwächer ein Signal um 20 dB abschwächen wollen? In beiden Fällen ergibt die Multiplikation ein Ergebnis in doppelter Genauigkeit mit einer Wortbreite von 48 bit. Das wollen wir kurz am vereinfachten 4-bit-DSP betrachten. Zu multiplizieren sind zwei Zahlen:

0,010 * 0,001 Zahlenwerte: 0,25 * 0,125 = 0,03125
0,000 0100 Zahlenwert: 0,03125

Das Ergebnis hat acht Stellen.
Einer der oben genannten Tipps soll am Beispiel veranschaulicht werden. Wenn man Integerzahlen miteinander multiplizieren muss, zum Beispiel, um Adressen für Speicherzugriffe zu berechnen, erhält man mit einem DSP-Multiplizierer ein Ergebnis, das mit einer Schiebeoperation angepasst werden muss. Wir bleiben beim zuletzt gerechneten Beispiel und interpretieren lediglich die Zahlen anders:

0010 * 0001 Zahlenwerte: 2 * 1 = 2
0000 0100 Zahlenwert: 4, das ist das Ergebnis des Fractional-Multiplizierers
0000 0010 Zahlenwert: 2, korrektes Ergebnis nach arithmetischem Rechtsshift um eine Stelle.

Demnächst...

So viel zum Aufbau der DSP-Programme. Im sechsten Teil werden wir das DSP-Board für einen digitalen Audio-Signalgenerator verwenden.

(110005)

First Step

NEU!

Erste Schritte mit dem Mikrocontroller

Sie interessieren sich als Auszubildender, Schüler, Student – oder einfach nur so – für Mikrocontroller-Technik? Mit dem neuen „First Step"-Paket haben Sie den Schlüssel und alle nötigen Werkzeuge für diese faszinierende Welt in der Hand! Das fertig bestückte und getestete „First Step"-Board und drei exakt darauf abgestimmte Arbeitshefte (plus Software-CD) machen die ersten Experimente mit einem Mikrocontroller zum Kinderspiel.

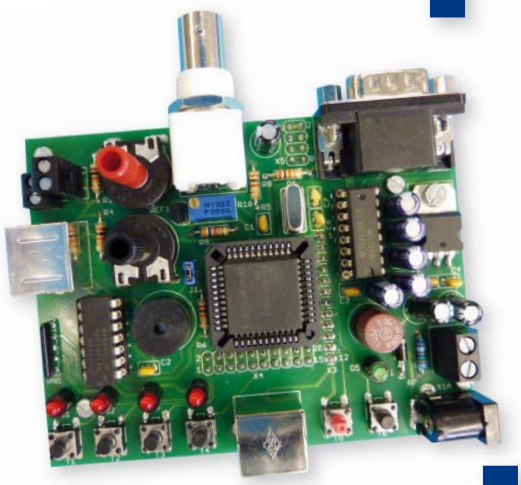
Bestandteile des „First Step"-Pakets:

elektor ACADEMY
the school of electronics

3 Arbeitshefte

(inkl. passendes DIN A4-Ringbuch)

- Beschreibung der Hardware
- Beschreibung der integrierten Entwicklungsumgebung IDE
- Einführung in die Programmiersprache „C"
- Zahlensysteme, Arithmetik, Variable
- Logische Operationen
- Digitale I/O-Ports
- A/D-Wandler, Tiner/Counter



1 „First Step"-Mikrocontroller-Board

- 8051er-Mikrocontroller: AT89C51CC03
- 2,5-V-Referenzspannungsgeber für A/D-Wandler: LT1009
- TTL/RS-232-Pegelwandler MAX232
- Treiber für LEDs und Piezo-Summer: 74HC04
- 4 Taster (Eingabe von binären Signalen)
- 4 LEDs (Ausgabe von binären Signalen)
- Piezo-Summer (Ausgabe von akustischen Signalen)
- BNC-Buchse (Ein-/Ausgabe von externen binären Signalen)
- 2 Potentiometer (Eingabe von analogen Signalen)
- 2 Mini-DIN-Buchsen und eine Doppelstock-Schraubklemme
- Karten-Format: 98 x 75 mm
- Spannungsversorgung: 9 V DC, max. 100 mA, Verpolungsschutzdiode und Miniatursicherung

1 CD-ROM mit Zusatzinfos

- Datenblätter
- Systemdokumentation
- Entwicklungsumgebung
- Beispielprogramme

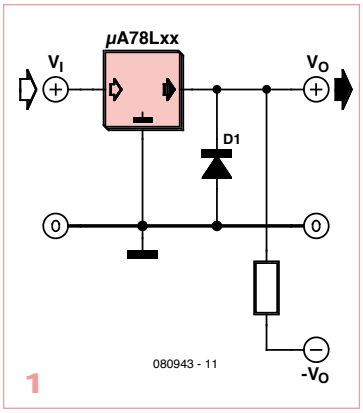
Das gesamte „First Step"-Paket kostet nur 199,00 Euro.

Weitere Infos und Bestellung unter
www.elektor.de/first-step

Sicherung für Spannungsregler

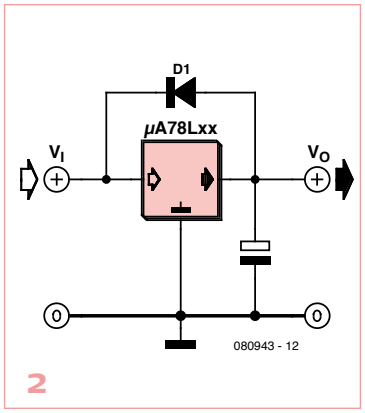
Von Ton Giesberts

Es kommt vor, dass die an Spannungsreglern angeschlossenen Lasten nicht gegen Masse, sondern gegen niedrigere Potentiale oder sogar gegen die negative Betriebsspannung liegen (wir betrachten positive Spannungsregler, für negative Typen gilt das Gleiche mit umgekehrten Vorzeichen). Diese Situation kann zum Beispiel zusammen mit Opamps oder Level-Shiftern auftreten. Eine Diode (1N4001 oder ähnlich) in Durchlassrichtung hinter dem Ausgang - wie in **Bild 1** dargestellt - schützt den Spannungsregler vor Beschädigung. Ohne eine solche Schutzdiode können Polaritätsvertauschungen, die zum Beispiel beim Einschalten oder bei Kurzschlüssen auftreten, einen Spannungsregler sofort zerstören.



Die Schutzdiode bewirkt, dass das Potential am Ausgang nicht unter Massepotential minus 0,7 V sinken kann. So geschützt, überleben kurzschlussfeste Spannungsregler wie die Typen der 78xx-Standardreihe diese Situation klaglos.

Es lässt sich nicht ausschließen, dass die Eingangsspannung eines Spannungsreglers schneller sinkt als seine Ausgangsspannung. Dieser Fall kann eintreten, wenn die Betriebsspannung aus Sicherheitsgründen kurzgeschlossen wird, weil am Ausgang eine Überspannung erkannt wurde. Wenn die Ausgangsspannung des Spannungsreglers um etwa 7 V über seiner Eingangsspannung liegt, bricht die Basis-Emitter-Strecke des internen Leistungstransistors durch. In diesem Fall wird der Spannungsregler sofort defekt. Dieser Gefahr kann man durch eine parallele Schutzdiode aus dem Weg gehen, die wie in **Bild 2** geschaltet ist. Die parallele Schutzdiode bewirkt, dass die hohe Ausgangsspannung zum Eingang kurzgeschlossen wird.



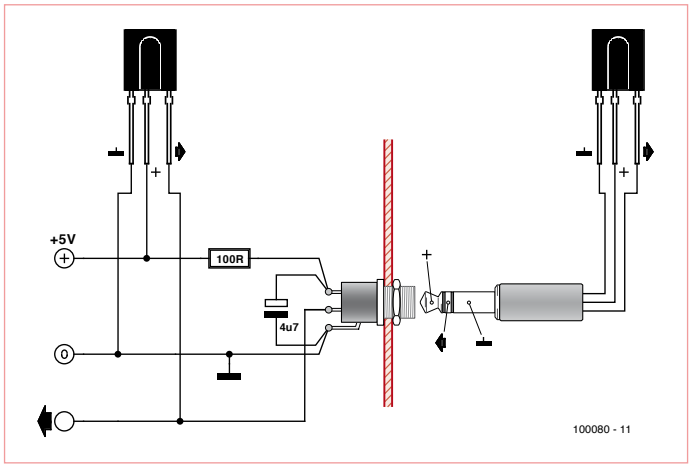
(080943)gd

Receiver-Fernbedienungs-Bereich erweitern

Von Géry Szczepanski

Wenn das TV-Gerät auf einem drehbaren Sockel steht, während der Satellitenreceiver seinen Platz ortsfest im Schrank hat, kann bei extremen Drehwinkeln der direkte Weg zum Infrarot-Empfänger unterbrochen sein. Der Receiver ist mit der Fernbedienung nur noch bedingt steuerbar. Das Problem lässt sich mit einem zweiten, parallelen Infrarot-Empfänger lösen, der an der Front des TV-Geräts montiert wird. Diese Idee ist auch auf andere Settop-Boxen wie DVB-T- oder DVB-C-Receiver übertragbar. Der Autor hat einen Satelliten-Receiver des Typs XSAT CD TV360 folgendermaßen nachgerüstet:

Die Platine wird aus ihren Halteclips genommen, so dass die Sensor-Anschlüsse +5 V, Masse und Signalausgang zugänglich sind. Ein Multimeter zeigte am Signalausgang die Spannung +4,5 V an. Die Sensor-Anschlüsse wurden an eine 3,5-mm-Klinkenbuchse auf der Receiver-Rückseite gelegt. Der 100-Ω-Widerstand in der +5-V-Leitung ist notwendig, weil der Klinkenstecker die Leitung beim Einführen vorübergehend nach Masse kurz schließt. Vom Klinkenstecker führt eine dreiadrige Leitung zum zweiten, am Rahmen des TV-Geräts befestigten Infrarot-Sensor. Nun kann der Receiver sogar hinter Möbeln oder in einer geschlossenen Schublade untergebracht werden. Eine ausreichende Belüftung muss natürlich gewährleistet sein.



(100080)gd

Subscribe to *audioXpress* magazine!

Do your **electronics speak** to you? Are the words **"audio", "vacuum tubes",** and **"speaker technology"** music to your ears?

Then you should be reading *audioXpress!*

Recently acquired by The Elektor Group, *audioXpress* has been providing engineers with incredible audio insight, inspiration and design ideas for over a decade. If you're an audio enthusiast who enjoys speaker building and amp design, or if you're interested in learning about tubes, driver testing, and vintage audio, then *audioXpress* is the magazine for you!

What will you find in *audioXpress*?

- In-depth interviews with audio industry luminaries
- Recurring columns by top experts on speaker building, driver testing, and amp construction
- Accessible engineering articles presenting inventive, real-world audio electronics applications and projects
- Thorough and honest reviews about products that will bring your audio experiences to new levels

Choose from print delivery, digital, or a combination of both for maximum accessibility.

Subscribe to *audioXpress* at
www.cc-webshop.com
today!

audioXPRESS

Sparsamer Spannungsmonitor

Von Rolf Blijleven (NL)

Mit Solarzellen kann man Elektronik einfach autonom versorgen, solange diese nicht dauerhaft in Betrieb sein muss. Ansonsten kann man jedes Quäntchen Strom sammeln, bis genug Energie vorhanden ist, die für den Betrieb einer Schaltung reicht. Doch wann ist „genug“? Ein Spannungsmonitor ist hier hilfreich, braucht allerdings selbst Energie. Er sollte daher so wenig Energie abzwacken wie möglich - je weniger µA desto besser!

In der Januar-Ausgabe von Elektor ging es um das Thema „Energy Harvesting“. Auch da wurde Elektronik an schwachen Energiequellen ohne Netz oder Batterien betrieben. In diesem Beitrag geht es um eine kleine Solarzelle der 5-€-Klasse, die bei 100 mA noch 1V Spannung liefert. Diesen Maximalwert erreicht sie allerdings nur im direkten Sonnenschein bei korrekter Ausrichtung. Ist es bewölkt, liefert sie bei einigen hundert mV gerade mal einige zig µA. Doch auch mit so wenig Energie kann man noch etwas anfangen, solange man keinen Dauerstrom braucht, sondern den Verbraucher nur kurz ein paar Mal am Tag einschaltet.

Anordnung

Bild 1 zeigt die Prinzipschaltung. Am Ausgang der Solarzelle hängt ein Spannungsvervielfacher, der den Speicherkondensator Cs auflädt. Spannungsvervielfacher-Schaltungen wurden im schon erwähnten Artikel im Januar ausgiebig besprochen, weshalb hier lediglich symbolisch ein Block dargestellt ist. Der die Spannung von Cs überwachende Komparator wird auch direkt aus Cs versorgt. In der realen Schaltung wird man hier auch noch einige Widerstände finden. Überschreitet die Spannung an Cs eine Schwelle, dann liefert der Komparator eine Cs entsprechende hohe Ausgangsspannung. In der Folge wird der MOSFET durchschalten und die Last an Cs legen. Das wird die Spannung an Cs wieder sinken lassen und somit kippt der Ausgang des Komparators wieder zurück. Durch die Diode wird Ct allerdings noch eine Weile Spannung führen und der Transistor durchgeschaltet bleiben. Die Zeit hängt von der Zeitkonstante aus Ct und Rt ab. Konstruktionen dieser Art werden auch als „Solar Engine“ bezeichnet. Im Internet findet man einiges zu diesem Thema.

An einem bewölkten Morgen kann man mit einem Solarstrom zwischen 20 und 30 µA

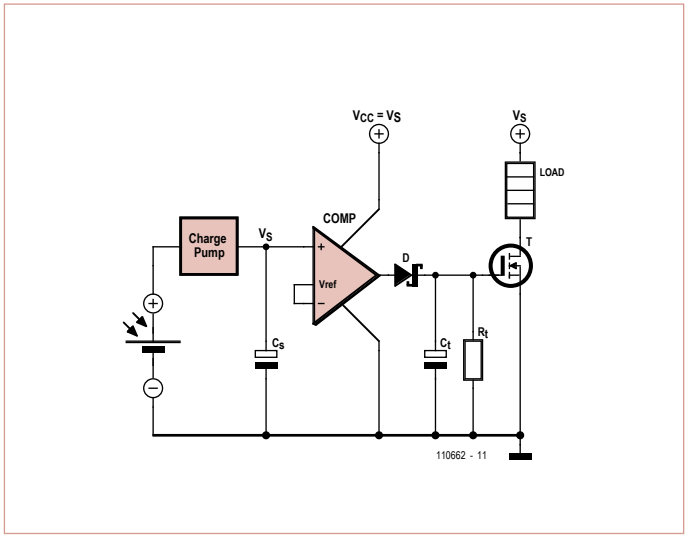


Bild 1. Prinzip-Schaltung der Elektronik zum Energy Harvesting.

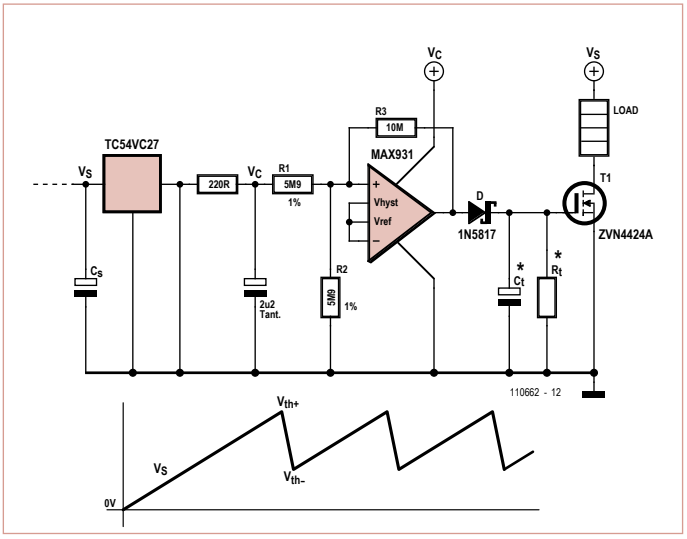


Bild 2. Schaltung des sparsamen Spannungsmonitors. Die Halbleiter sind sowohl bedrahtet als auch als SMD erhältlich.

Leserprojekte sind Beiträge von Elektor-Lesern für experimentelle Zwecke oder zur Anregung für andere Leser.
Die in dieser Rubrik vorgestellten Schaltungen wurden vom Elektor-Labor nicht auf Reproduzierbarkeit und Funktion getestet.

rechnen. Cs lädt sich also sehr langsam auf. Dennoch muss man kontinuierlich überwachen, ob schon genug Energie in Cs steckt. Ein einfacher Timer würde nicht genügen. Experimente mit dem Spannungsmonitor MAX8212 von Maxim [1, 2] als Komparator waren nicht wirklich befriedigend. Das IC verbraucht nämlich zusammen mit einigen Einstellwiderständen schon 6 µA und vergeudet damit schon ein Viertel der bei bewölktem Himmel zur Verfügung stehenden Energie.

Gute Lösung

Besser klappt das mit der Schaltung von **Bild 2**. Der dreipolige Spannungsdetektor TC54VC27 benötigt <1 µA, solange die Schwellenspannung von 2,7 V noch nicht erreicht ist. Dies dürfte die meiste Zeit der Fall sein.

Oberhalb der Schwelle schaltet das IC die Spannung vom Eingang zum Ausgang durch. Nun wird der Komparator MAX931 (Strombedarf 4 µA) mit Spannung versorgt. Bei einem schlagartigen Einschalten kann sein Ausgang aber einen kurzen Spannungsimpuls liefern, der den Schalttransistor durchsteuern würde. Anschließend würde Vs einbrechen und Cs kommt so nicht recht auf die gewünschte Spannung. Um dieses Verhalten zu vermeiden, wird das Einschalten mit einem RC-Netzwerk aus 220 Ω und 2,2 µF besänftigt. Das IC MAX931 bleibt so unterhalb der Schaltschwelle Vth+, die durch R1, R2 und R3 mit den angegebenen Werten auf etwa 3,1 V festgelegt wurde. Der Unterschied zu den 2,7 V ist zwar gering, aber entscheidend. Das merkt man vor allem dann, wenn die Last aus einem kleinen Motor besteht. Die Berechnung von R1 und R2 entnimmt man dem zugehörigen Datenblatt [3]. Im Bereich zwischen 2,7 und 3,1 V bleibt der Strombedarf der Schaltung bei 5 µA.

TC54-Varianten gibt es mit Schwellen von 1,4 bis 7,7 V (siehe Datenblatt [4]). Man nehme also ein Exemplar mit einer Schwellenspannung knapp unterhalb der gewünschten Vth+. Falls Sie sich fragen, ob der zusätzliche Komparator tatsächlich notwendig ist - die Antwort ist: ja! Der Transistor muss nämlich definiert eingeschaltet werden. In Bild 2 schaltet der TC54 bei 2,75 V. Ist der MOSFET direkt hinter den TC54 geschaltet, dann wird er leitend, die Spannung über Cs bricht ein und der TC54 schaltet wieder aus…

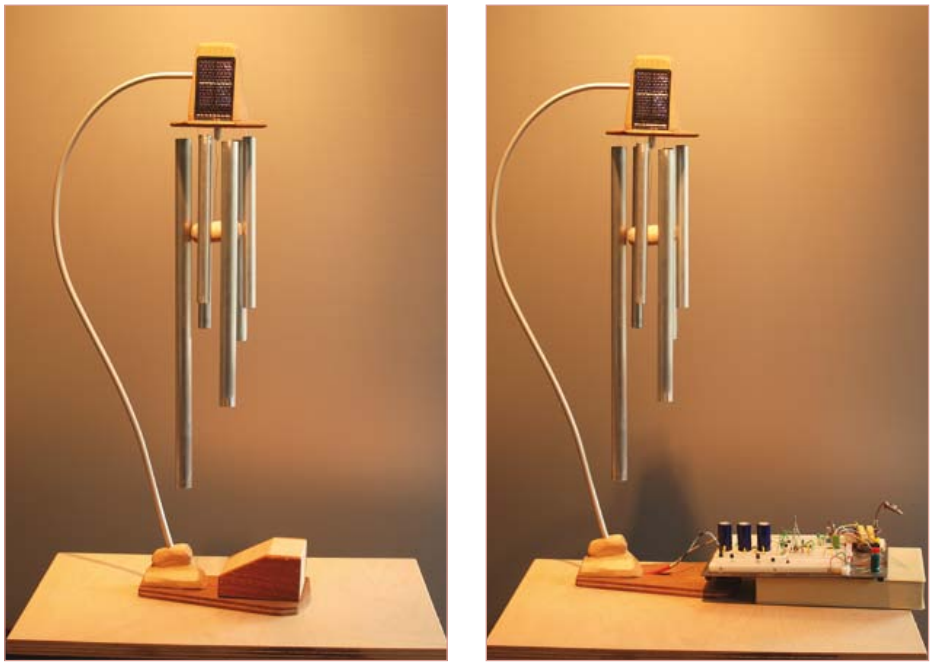


Bild 3. Ein Glockenspiel als akustischer Kondensator-ist-voll-Melder.

Auch ein Mitkoppelwiderstand vom Aus- auf den Eingang ist keine gute Lösung, denn der würde Cs zu sehr belasten.
Die Zeitkonstante aus Ct und Rt bestimmt die Durchschaltdauer des Transistors. Cs muss nicht voll entladen werden, denn das Ausschalten übernimmt die Last. Wenn die Belastung zusammen mit R_{dsON} des MOSFETs zum Beispiel bei 25 Ω liegt und Cs aus drei Elkos mit je 4.700 µF aufgebaut ist, dann wird Cs in rund 0,35 s entladen sein. Wenn aber eine Betriebsdauer von 0,2 s genug ist, dann kann Cs zu einem Drittel gefüllt bleiben. Die verkürzte Einschaltdauer erreicht man bei Ct = 2,2 µF mit Rt = 100 kΩ. Der Verlauf der Spannung Vs über die Zeit ist in der Grafik dargestellt.

Kleine Unterschiede können relevante Auswirkungen haben – auf jeden Fall kann man die Unterschiede klar erkennen. Es macht allerdings wenig Sinn, beim Testen solch lang dauernder Prozesse immer wieder auf ein Voltmeter zu schauen. Aus diesem Grund bestand die Last beim Autor aus einem elektromotorisch betriebenen Glockenspiel. Er konnte so hören, wann der Kondensator geladen ist (und damit auch das Wetter draußen „erhören“). Mit dem sparsamen Spannungsmonitor läutet es auch bei Regenwetter ein paar Mal am Tag. Weniger optimierte Schaltungen bleiben da still.

(110662)

Weblinks

[1] http://library.solarbotics.net/circuits/se.html

[2] www.maxim-ic.com/datasheet/index.mvp/id/1273

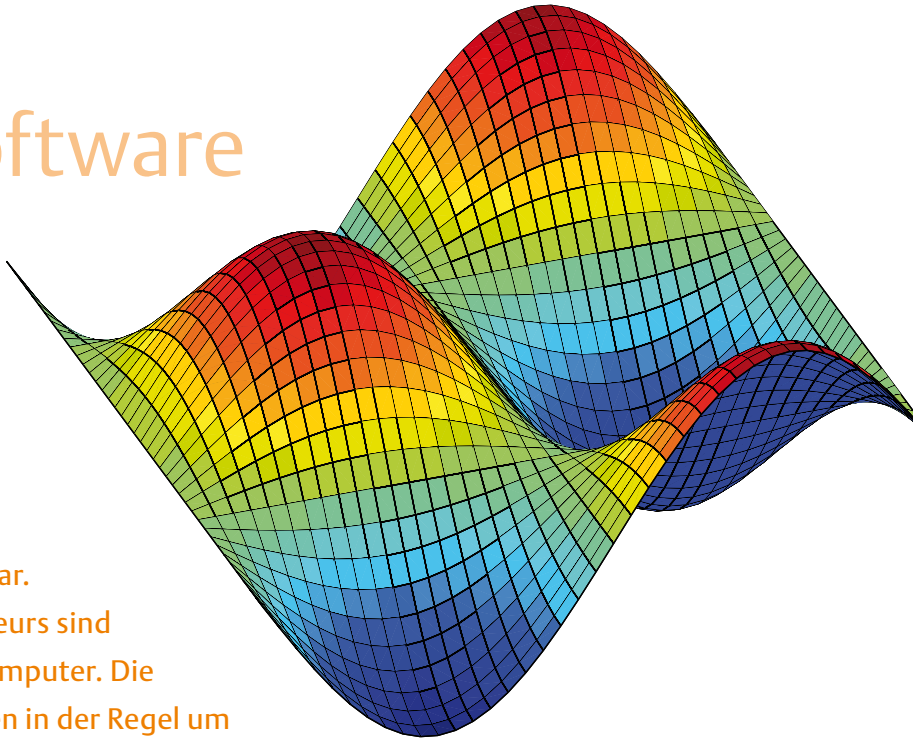
[3] www.iamwhen.com/archives/53-Herbert-1701-Species-B-Generation-1.html

[4] www.maxim-ic.com/datasheet/index.mvp/id/1219

[5] www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010716

Scilab

Open-Source Software für numerische Berechnungen



Von Vincent Couvert, Bruno Jofret und Julie Paul (F)

Zum Lösen mathematischer Aufgaben und Problemstellungen ist Software unverzichtbar. Bekannte Beispiele aus der Praxis des Ingenieurs sind Modellrechnungen und Simulationen am Computer. Die Kosten realer Tests und Versuchsreihen liegen in der Regel um ein Vielfaches höher. Schwerpunkte in der Anwendung numerischer Verfahren sind die Automobil- und Flugzeugindustrie, die Energieerzeugung und Energieverteilung, die chemische Industrie und nicht zuletzt die Welt der Börsen und Banken.

In der Industrie, gleich welcher Sparte, ist der Konkurrenzdruck enorm. Die Gesetze des Marktes sind hart, deshalb sind permanente Kostensenkungen überlebenswichtig. Auch der Bilanzposten Simulationssoftware bleibt von Kürzungen und Streichungen nicht verschont. Vor noch nicht allzu langer Zeit war Simulationssoftware für industrielle Anwendungen eine proprietäre Domäne. Die Urheber gaben die Rechte an ihren Produkten nicht aus der Hand, Anpassungen an Kundenwünsche waren ziemlich teure Maßarbeit. Mit Erscheinen quelloffener Pakete haben diese Software-Häuser ernst zu nehmende Mitstreiter bekommen.

Noch vor einigen Jahren war das breite Publikum die Hauptzielgruppe quelloffener Programme, zum Beispiel bei Web-Browsern und Textsystemen. Inzwischen bedienen sich auch öffentliche Verwaltungen aus diesem Fundus, und in die Forschung und Industrie ist die Open-Source-Software ebenfalls vorgedrungen. Scilab [1], hinter dem ein Konsortium industrieller Anwender steht, ist zu einer vollwertigen Alternative zum „Industriestandard“ Matlab [2] geworden. Ein nicht zu widerlegendes Argument dürfte neben dem Kostenvorteil der uneingeschränkte Zugang zu den Programmquellen sein. Scilab wird heute im militärischen Bereich ebenso eingesetzt wie in der zivilen Luft- und Raumfahrt. In Frankreich wurde Scilab an technischen Ausbildungseinrichtungen zum fast schon selbstverständlichen Handwerkzeug. Unlängst erkannte das französische Bildungsministerium Scilab offiziell als Lehrmittel an.

Scilab in der Praxis

Scilab steht unter der CeCILL-Lizenz, die mit der GPL-Lizenz vergleichbar ist. Das Programmpaket ist für die bekannten Betriebssysteme (Windows, Mac, Linux) verfügbar, alle Versionen können kostenlos von der Scilab-Website [1] heruntergeladen werden.

Scilab ist eine in sich abgerundete Umgebung für mathematische Berechnungen, wissenschaftliche Modelle, Simulationen und Visualisierungen. Ursprünglich zum Berechnen von Matrizen entwickelt, umfasst Scilab heute hunderte mathematische Funktionen sowie eine leistungsstarke Programmiersprache. Die Anbindung an andere, in verbreiteten Sprachen geschriebene Programme (C, C++ oder Java) ist möglich. Die Syntax von Scilab stimmt mit der von Matlab weitgehend überein, absolut kompatibel sind Scilab und Matlab jedoch nicht. Das bereits sehr ausgedehnte funktionale Spektrum kann vom Anwender fast beliebig erweitert werden. So können beispielsweise Module für die Simulation, die grafische Visualisierung, die Optimierung, für statistische Aufbereitungen, Systementwicklungen und Systemanalysen, für die Signalverarbeitung sowie vieles andere mehr ergänzt werden.

Ebenso wie die Mehrzahl anderer quelloffener Programme ist Scilab interoperabel, was bedeutet, dass der Anwender bei der Modifikation und Konfiguration freie Hand hat. Als Beispiel sei LabVIEW [3] von National Instruments genannt: Mit einem Gateway zwischen LabVIEW und Scilab lässt sich eine vollwertige und leistungsstarke Verarbeitung der Daten realisieren. Der Anwender schreibt sein Scilab-Skript in LabVIEW und setzt Scilab für die Analyse und Visualisierung seiner Daten ein.

Integrierte Umgebung

Wer schon einmal mit einer grafischen Programmiersprache unter einem Interpreter gearbeitet hat, wird mit der Entwicklungsumgebung von Scilab schnell vertraut sein. Die Wirkungen der in die Konsole eingegeben Kommandos werden sofort sichtbar. Mit SciNotes, dem eingebauten Texteditor, können Funktionen zu Skripten zusammengefasst und gespeichert werden, so dass die folgende

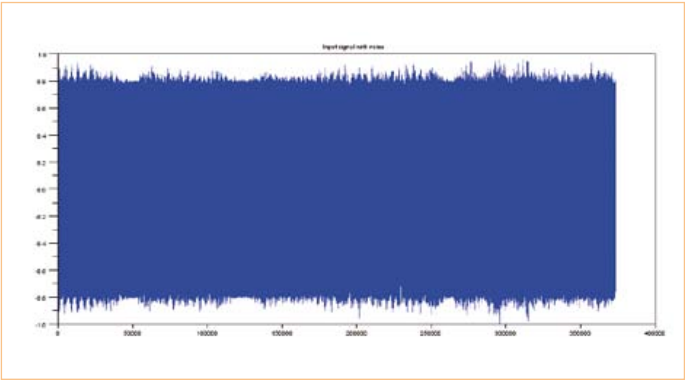


Bild 1. Das Audio-Eingangssignal.

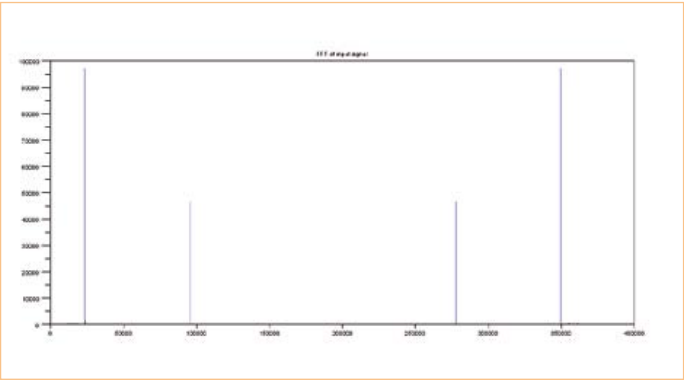


Bild 2. Fourier-Transformation des Eingangssignals, das Störungen auf zwei diskreten Frequenzen enthält.

Sitzung in gleicher Arbeitsumgebung anschließt. SciNotes ist in die Konsole von Scilab integriert. So können beispielsweise Funktionsgruppen ganz oder teilweise ausgeführt werden, ebenso wie in einer Programmierumgebung.

Visualisierungsfunktionen

SciLab hält für die zwei- und dreidimensionale Dateninterpretation grafische Funktionen bereit, die Resultate können kommentiert und exportiert werden. Die Auswahl an Grafik- und Diagramm-Formaten ist reichhaltig. Vom Anwender generierte Daten sind in der Scilab-Konsole interaktiv präsentierbar.
Nachfolgend wollen wir den Umgang mit Scilab an zwei Beispielen demonstrieren, die Bezug zur elektronischen Signalverarbeitung haben.

Erstes Beispiel: Audiofilter

Dieses Beispiel soll zeigen, wie ein akustischer Klang, gespeichert im WAV-Format, mit Scilab durch ein Filter geschickt werden kann. In Scilab sind die Funktionen für alle notwendigen Schritte enthalten: Verwalten der Dateien, schnelle Fourier-Transformation, Berechnen und Anwenden digitaler Filter.
Der Name der Datei, um die es in diesem Beispiel geht, ist `noisySignal.wav` [4]. Wir speichern diese Datei im aktuellen Ordner von Scilab, mit der Funktion `cd` lässt sich der aktuelle Ordner ändern. Die Funktion `loadwave` öffnet `noisySignal.wav` in Scilab. Intern zeigt ein Vektor auf die Signaldaten, ergänzt durch Informationen zum Datei-Inhalt, zur Sample-Frequenz und zu weiteren Parametern. Über SciNotes geben wir folgende Kommandos ein, wir lassen sie anschließend durch *Execute → ..until the caret, with echo* ausführen:

```
stacksize(„max“);  // Maximaler Speicher für Scilab
[noisySignal, noisySignalInfo] =
loadwave(„noisySignal.wav“);
sampleFrequency = noisySignalInfo(3);
samples = noisySignalInfo($);
```

In der Konsole erscheint folgende Antwort:

```
-->sampleFrequency = noisySignalInfo(3)
 sampleFrequency  =

    22050.

-->samples = noisySignalInfo($)
 samples  =

    373380.
```

Dies bedeutet, dass die Sample-Frequenz 22050 Hz beträgt und 373380 Samples vorhanden sind. Mit `plot` lässt sich das Signal, wie **Bild 1** zeigt, visualisieren:

```
plot(noisySignal)
xtitle(„Input signal with noise“);  // Titel der
Grafik
```

Um das Nutzsignal vom Rauschen zu trennen, wenden wir mit der Funktion `fft` die Fourier-Transformation an. Gleichzeitig lassen wir den Absolutwert mit `abs` berechnen:

```
FftOfNoisySignal = abs(fft(noisySignal));
scf();  // Öffne ein grafisches Fenster
plot(FftOfNoisySignal);
xtitle(„FFT of input signal“)
```

Die Funktion `plot` liefert die Grafik, die in **Bild 2** wiedergegeben ist. Dort ist deutlich erkennbar, dass in dem symmetrischen Signal zwei Störungen enthalten sind. Die Störungen müssen in zwei Schritten aus dem Signal entfernt werden. Zuerst lassen wir von Scilab mit den integrierten Matrix-Funktionen die Frequenzen der Störungen berechnen. Scilab gibt die Werte der Spitzen und ihre Orte auf der

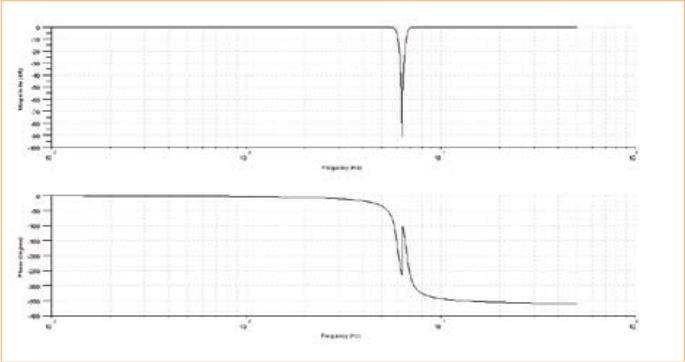


Bild 3. Bode-Diagramm des IIR-Filters, das die Störung bei 1397 Hz unterdrückt.

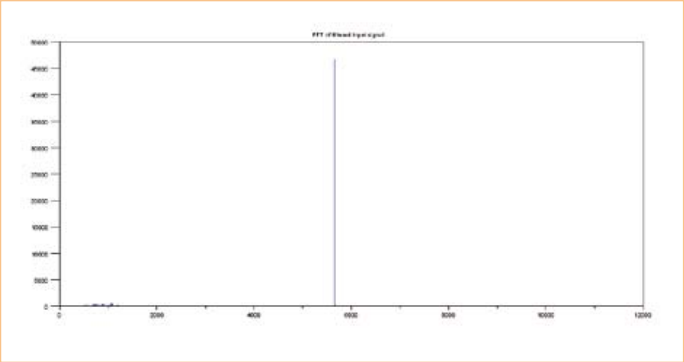


Bild 4. Nach der ersten Filterung ist nur noch eine Störung vorhanden.

Frequenzachse zurück, wenn wir folgende Kommandos eingeben:

```
// Die Fourier-Transformation ist symmetrisch,
// deshalb genügt der erste Mittelwert
frequencies = sampleFrequency*(0:(samples/2))/
samples;

// Bestimme die Frequenzen der zu filternden Spitzen
[peakValue, peakValueIndex] = max(FftOfNoisySignal(1:
size(frequencies, „*“)));
peakValueIndex
frequencies(peakValueIndex)
```

```
Das Ergebnis erscheint in der Konsole:

-->peakValueIndex
 peakValueIndex  =

    23657.

-->frequencies(peakValueIndex)
 ans  =

    1397.0079
```

Die erste Spitze ist eine Störung auf 1397,0079 Hz. Wir eliminieren diese Störung mit einer Bandsperre auf dieser Frequenz (Butterworth 3. Ordnung), die Bandbreite soll 200 Hz betragen. Mit der Funktion `iir` legen wir die Übertragungsfunktion des Filters fest:

```
hz = iir(3, „sb“, „butt“, [frequencies(peakVal
ueIndex)-100 frequencies(peakValueIndex)+100]/
sampleFrequency, [0 0]);
```

Das Bode-Diagramm des Filters (**Bild 3**) ist mit der Funktion `bode` abrufbar. Anschließend genügt eine einzige Kommandozeile, damit Scilab die Filterung ausführt:

```
// Filterung des Eingangssignals
filteredSignal = filter(hz.num, hz.den,
filteredSignal);
```

Nachdem die erste Störung, wie das Fourier-Diagramm in **Bild 4** beweist, durch den Butterworth-Filter 3. Ordnung beseitigt wurde, muss noch die zweite Störung gefiltert werden. Diese Störung eliminieren wir mit einem Chebyshev-Filter Typ I, 3. Ordnung. Im Übrigen ist die Vorgehensweise mit dem vorangegangenen ersten Schritt identisch:

```
// Bestimme die Frequenz der zu filternden Spitze
[peakValue, peakValueIndex] = max(FftOfNoisySignal(1:
size(frequencies, „*“)));
peakValueIndex
frequencies(peakValueIndex);

// Stelle für diese Frequenz ein Chebyshev-Filter ein
hzFiltre2 = iir(3, „sb“, „cheb1“, [frequencies(pea
kValueIndex)-100 frequencies(peakValueIndex)+100]/
sampleFrequency, [0.01 0]);

// Filtere das Signal in einem Durchgang
filteredSignal = filter(hz.num, hz.den,
filteredSignal);
```

Nach Eingabe folgender Kommandos erscheint das in **Bild 5** dargestellte Ergebnis:

```
scf();
plot(filteredSignal)
xtitle(„Input signal filtered twice“)
```

Damit ist die Filterung abgeschlossen. Mit einer letzten Fourier-Transformation können wir uns davon überzeugen, dass beide Störungen eliminiert sind (**Bild 6**):

```
FftOfFilteredSignal = abs(fft(filteredSignal));
scf();
plot(frequencies, FftOfFilteredSignal(1:size(frequenc
ies, „*“)));
xtitle(„FFT of input signal filtered twice“)
```

Das Ergebnis unserer Arbeit speichern wir als Datei im WAV-Format:

```
savewave(„SoundFiltered.wav“, filteredSignal)
```

Zweites Beispiel: Kantenfilter

In diesem Beispiel begeben wir uns auf das Gebiet der elektronischen Bildverarbeitung. Genauer beschrieben geht es um eine Pro-

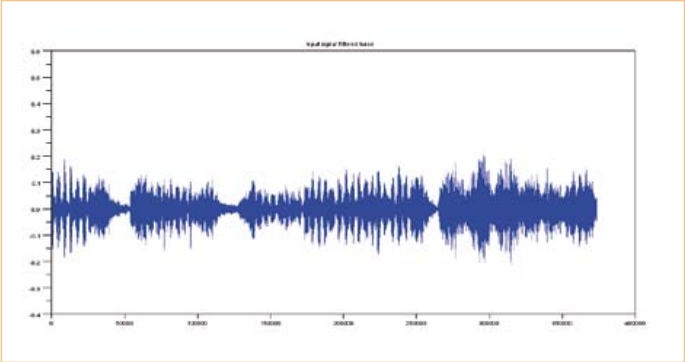


Bild 5. Ergebnis der zweiten Filterung: Beide Störungen sind eliminiert.

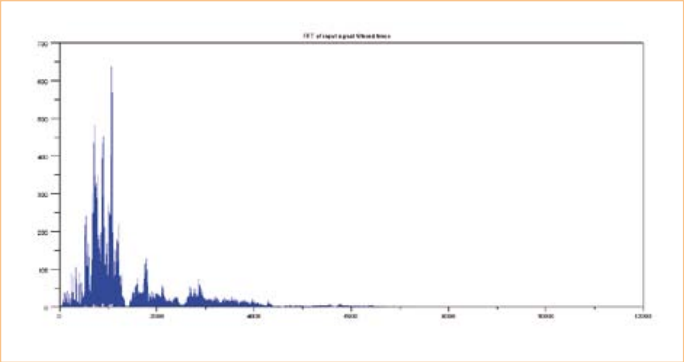


Bild 6. Das gefilterte Signal nach der Fourier-Transformation.

blemstellung, die am Anfang einer elektronischen Bilderkennung steht. Das Erkennen von Kanten oder Umrissen eines Objekts oder einer Person ist der erste komplexe Schritt auf dem Weg zur Identifizierung von Gegenständen oder Gesichtern (face tracking).
Das Bild, das verarbeitet werden soll, ist als Matrix aus Werten von Helligkeitsabstufungen interpretierbar. Scilab übernimmt hier zwei Aufgaben: Es stellt Funktionen zum Berechnen von Matrizen bereit, und es macht das Bild in den verschiedenen Verarbeitungsphasen sichtbar.
Wir wollen in diesem Beispiel von einem Gradienten-Verfahren Gebrauch machen und Prewitt-, Sobel- und Scharr-Filter anwenden. Zuerst öffnen wir in Scilab mit `readimage` eine Bilddatei, die den Namen Scilab.pgm [4] trägt und im PGM-Format vorliegt (PGM = Portable Gray Map, Bild in Grauwerten). Die Funktion lädt die Datei, stellt die Bildlänge und Bildbreite fest und ordnet jedem Pixel entsprechend seinem Grauton einen Zahlenwert im Bereich 0...255 zu. Das Ergebnis hat die Form einer Matrix:

```
stacksize(„max"); // Maximaler Speicher für Scilab
gray_m = readimage(„Scilab.pgm");
```

Der Einfachheit wegen benutzen wir in diesem Beispiel stets die Funktion `showImage`, mit ihr können wir die Ergebnisse der diversen Transformation betrachten. Diese Funktion geht auf `Matplot` zurück, eine Matrixfunktion, die zu den Grundsteinen von Scilab gehört. Das Bild wird mit den Werten der Grautontabelle aufgebaut:

```
function []=showImage(imageMatrix)
   f = Figure();
   f.color_map = graycolormap(255);
   f.background = -2;
   Matplot(imageMatrix);
  endfunction

 showImage(gray_m);
```

Nach Ausführen vorstehender Kommandos wird das geladene Bild in einem separaten grafischen Fenster (**Bild 7**) wiedergegeben. Für die Kantenerkennung in einem Grautonbild genügt eine zweidimensionale Matrix (Höhe und Breite). Da Scilab n-dimensionale Matrizen verarbeiten kann, ist auch die Filterung farbiger Bilder möglich (RGB, RGBA, CMYK, HSV und andere Formate). Die Farbkanäle werden einzeln verarbeitet, anschließend werden die Ergebnisse verglichen und zusammengeführt.

Bevor wir die Kantenfilterung durchführen, stellen wir unser Grautonbild etwas „flacher" ein. Dadurch werden Störungen wie Rauschen und Nadelspitzen gemindert, was die Zuverlässigkeit der Kantenerkennung steigert. Diese Vorverarbeitung führen wir in Scilab mit einer Konvolution (Faltung) und einer zweidimensionalen Gauss-Elimination durch. Die Eingangsgröße ist das Bild in Gestalt der Matrix, die Ausgangsgröße ist das vorverarbeitete Bild.

Mit der Funktion `sum` fügen wir dem Bildrand zwei Pixelreihen hinzu, eine Reihe für das Konvolutionsprodukt, die zweite Reihe für das Matrixprodukt. Da in Scilab optimierte Funktionen eingebaut sind, verläuft dieser Schritt wesentlich schneller als bei der Verarbeitung jedes einzelnen Elements:

```
function N=blurr(P)
    N = zeros(P);
    P2 = [zeros(1,size(P,"c")); zeros(1,size(P,"c"));
P; zeros(1,size(P,"c")); zeros(1,size(P,"c"))]
    P2 = [zeros(size(P2, „r"), 1), zeros(size(P2,
„r"), 1), P2, zeros(size(P2, „r"), 1), zeros(size(P2,
„r"), 1)]
    K = 1/159 * [2  4  5  4 2
    4  9 12  9 4
    5 12 15 12 5
    4  9 12  9 4
    2  4  5  4 2];

    for x=3:(size(P2, „r") - 2)
        for y=3:(size(P2, „c") - 2)
            r = 0;
            N(x-2,y-2) = sum(K .* P2(x-2:x+2,
y-2:y+2));
        end
    end
endfunction
```

In der Literatur sind einfachere Implementierungen der Konvolution zu finden, die zwar ein gleichwertiges Ergebnis liefern, jedoch wesentlich mehr Rechenzeit erfordern:

```
function N=dummy_blurr(P)
    N = zeros(P);
    K = 1/159 * [2  4  5  4 2
    4  9 12  9 4
```

Bild 7. Originales Bild in Grautönen.

Bild 8. Für die Kantenerkennung vorverarbeitetes Bild.

```
    5 12 15 12 5
    4  9 12  9 4
    2  4  5  4 2];
    for x=1:size(P, „r")
        for y=1:size(P, „c")
            r = 0;
            for i = -2:2
                for j = -2:2
                    if (x + i > 0 & x + i <= size(P,
„r") & y + j > 0 & y + j <= size(P, „c"))
                        r = r + K(i+3, j+3) * P(x+i,
y+j)
                    end
                end
            end
            N(x,y) = r;
        end
    end
endfunction
```

Die vorverarbeitete Version (**Bild 8**) wirkt etwas unscharf, was nicht etwa am Druck dieser Elektor-Ausgabe liegt. Von dem Zwischenprodukt lassen wir auf drei Wegen den Gradienten berechnen, dies ist die eigentliche Kantenerkennung. Die Gradienten-Berechnung führen wir mit einem Sobel-Filter und einem Prewitt-Filter durch, hierzu gehören die **Bilder 9** und **10**. Die Filter werden mit einer Funktion realisiert, die ein Konvolutionsprodukt zum Ergebnis hat:

```
function N=convol2d(K, P)
    N = zeros(P);
    P2 = [zeros(1,size(P,"c")); P;
zeros(1,size(P,"c"))]
    P2 = [zeros(size(P2, „r"), 1), P2, zeros(size(P2,
„r"), 1)]
    for x=2:(size(P2, „r") - 1)
        for y=2:(size(P2, „c") - 1)
            r = 0;
            N(x-1,y-1) = sum(K .* P2(x-1:x+1,
y-1:y+1));
        end
    end
endfunction
```

Andere Open-Source Software für numerische Berechnungen

Unter der freien Software gibt es neben Scilab weitere Programme mit verwandter Zielrichtung:

– **Octave** ist eine strukturierte Programmiersprache innerhalb der C-Familie. Ebenso wie Scilab arbeitet Octave mit einem Interpreter. Unterstützt werden die meisten Module der Standardbibliothek von C. Octave ist durch Unix-Funktionen erweiterbar, das Gleiche gilt für Funktionen, die in C++ geschrieben sind. Die meisten Kommandos stimmen ebenso wie die Syntax mit Matlab überein. Skripte laufen normalerweise ohne Anpassungen sowohl in Octave als auch in Matlab (Quelle: Wikipedia). *www.gnu.org/software/octave/*

– **FreeMat** ist eine Analyse-Umgebung und Programmiersprache in Gestalt einer freien Bibliothek, die mit dem Quellcode von Matlab und Octave mehr oder weniger kompatibel ist. An FreeMat kann externer, in C, C++ oder Fortan geschriebener Code angebunden werden. Damit ist FreeMat zum Entwickeln paralleler distributiver Algorithmen gut geeignet. Verfügbar sind einige Methoden zum Volume Rendering (Darstellen dreidimensionaler Objekte in der Ebene) und zur dreidimensionalen Visualisierung. Die letzte Version erschien im Oktober 2009 (Quelle: Wikipedia). *http://freemat.sourceforge.net*

– **JMathLib** kann als Java-Doppelgänger von Scilab, Octave, FreeMat und Matlab betrachtet werden. Anders als bei Scilab und Octave, jedoch ebenso wie bei FreeMat waren in den letzten Jahren keine Weiterentwicklungen zu beobachten. Die letzte Version 0.9.4 datiert vom Februar 2009. *www.jmathlib.de/*

Dem Konvolutionsprodukt entnehmen wir anschließend den Gradienten. Für einen Sobel-Filter erhalten wir:

```
GX = convol2d([-1 0 1 ; -2 0 2 ; -1 0 1], gray_m);
GY = convol2d([-1 -2 -1 ; 0 0 0 ; 1 2 1], gray_m);
contourSobel = sqrt(GX.^2+GY.^2);
showImage(contourSobel);
```

```
Für einen Prewitt-Filter sieht dies wie folgt aus:
```

```
GX = convol2d([-1 0 1 ; -1 0 1 ; -1 0 1], gray_m);
GY = convol2d([-1 -1 -1 ; 0 0 0 ; 1 1 1], gray_m);
contourPrewitt = sqrt(GX.^2+GY.^2);
showImage(contourPrewitt);
```

Die Konvolutionsmodelle von Costella, Robert Cross und Scharr liefern im Prinzip bessere Ergebnisse, doch dazu müssen wir andere Gleichungen anwenden. Die Syntax in Scilab entspricht fast der Schreibweise in der Mathematik, so dass sich Gleichungen recht elegant implementieren lassen: Variable müssen nicht deklariert werden, sie müssen folglich nicht bestimmten Typen angehören, das Zuweisen von Speicherplatz entfällt ebenfalls.
Nachdem wir so weit fortgeschritten sind, erstellen wir mit wenigen Kommandos ein Scharr-Filter (**Bild 11**):

```
GX = convol2d([3 0 -3 ; 10 0 -10 ; 3 0 -3], gray_m);
GY = convol2d([3 10 3 ; 0 0 0 ; -3 -10 -3], gray_m);
contourScharr = sqrt(GX.^2+GY.^2);
showImage(contourScharr);
```

Diese Matrix kann als Eingangsgröße von Algorithmen dienen, die Formen und Umrisse unterscheiden. In Richtung Bildverarbeitung ist dies natürlich nur der erste Schritt, eine Filterung mit Schwellenwerten als Kriterium könnte sich anschließen. Eine solche Filterung würde dazu führen, dass nur die Kanten der hellen Bildbereiche übrig bleiben. In diesem Fall enthält die resultierende Matrix nur die Werte 0 oder 1 für die Pixel, die zu einer Kante gehören.

Scilab ist natürlich nicht nur zur Filterung von Klängen und Bildern einsetzbar. In einer folgenden Elektor-Ausgabe wollen wir uns mit Xcos befassen, ein Scilab-Tool zum Modellieren und Simulieren hybrider dynamischer Systeme, vergleichbar mit Simulink in Matlab.

(110491)gd

Weblinks

[1] Scilab: www.scilab.org

[2] Matlab: www.mathworks.com/matlab

[3] LabVIEW: www.ni.com/labview

[4] Download zu diesem Beitrag: www.elektor.de/110491

[5] Hilfe zur Funktion Matplot: http://help.scilab.org/docs/current/fr_FR/Matplot.html

Bild 9. Kantenerkennung mit einem Sobel-Filter,

Bild 10. ...mit einem Prewitt-Filter,

Bild 11. ...mit einem Scharr-Filter.

SPICE it up
Simulieren mit LTspice

Von Raymond Vermeulen (Elektor-Labor)

Das Simulieren projektierter elektronischer Systeme hat sich unabhängig von ihrem Umfang als überaus nützlich erwiesen. So lassen sich beispielsweise Messergebnisse verifizieren und Anpassungen können ohne viel Mathematik vorgenommen werden. Zum Vorschein kommen auch nicht ideale Eigenschaften der Bauelemente, die in Formeln keine Berücksichtigung finden. Kurze und bündige Hilfestellung für Leser, die mit SPICE-Simulatoren noch nicht in Berührung gekommen sind, gibt dieser Beitrag.

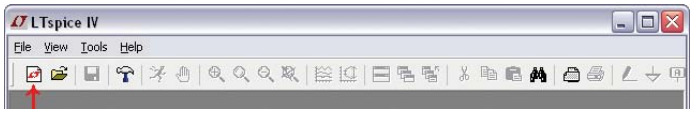
LTspice ist ein Simulationsprogramm von Linear Technology (LT), das SPICE zur Grundlage hat und von LTs Website kostenlos und ohne Registrierung heruntergeladen werden kann. Ein kleiner Wermutstropfen liegt darin, dass die Bauteilbibliotheken ausschließlich Produkte von LT enthalten. LTspice eignet sich daher besonders gut zum Simulieren von Systemen, die mit Bauelementen von LT arbeiten. Vom Anwender können jedoch Erzeugnisse fremder Hersteller hinzugefügt werden.
Auf seiner Webseite demonstriert LT die Fähigkeiten an zahlreichen Beispielen. Stark erweiterte Funktionalitäten und noch wesentlich mehr Bauteilbibliotheken bieten Simulatoren wie Micro-Cap oder Orcad Pspice, doch sie gibt es leider nicht zum Nulltarif. Ein wichtiger Grund, LTspice zu probieren, ist der fast spielerische Einstieg in die Welt der Simulatoren: Wenn der Entwickler mit einem auf SPICE basierenden Simulator vertraut ist, stellen andere Simulatoren keine große Hürde dar.

Anfänge

Der Grundstein für SPICE wurde in den siebziger Jahren des vergangenen Jahrhunderts an der University of California gelegt. Auftraggeber war das Verteidigungsministerium der USA, das die Strahlungsresistenz elektronischer Bauelemente erforscht wissen wollte. In den folgenden Jahren kamen zahlreiche Verbesserungen und Erweiterungen mit dem Ziel hinzu, auch hoch komplexe Bauelemente und Funktionen simulieren zu können. Noch später erschienen kommerzielle Simulatoren, die auf SPICE aufbauten und sich mit einer eigenen Benutzeroberfläche präsentierten.

Starten mit LTspice

Besuchen Sie die Website von LT (www.linear.com) und wählen Sie *Design Support*, folgen Sie unter *Design Simulation* dem Link *Design Simulation Page*. Von der nächsten Seite können Sie LTspice herunterladen. Starten Sie die heruntergeladene Datei und installieren Sie das Programm in einen Ordner Ihrer Wahl. Unter Linux läuft LTspice zusammen mit dem Windows-Emulator *Wine*, unter Ubuntu 11.04 wurde dies getestet.
Starten Sie LTspice und legen Sie ein neues Arbeitsblatt an, indem Sie auf das Symbol links außen klicken:



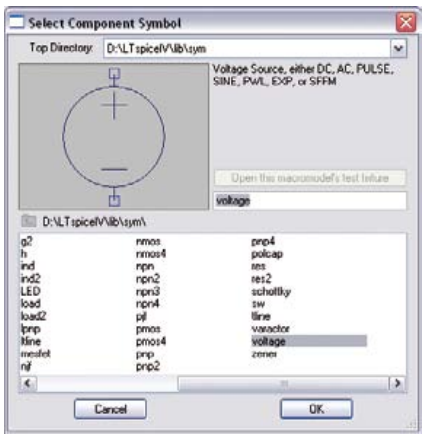


Bild 1. Wahl der Komponenten.

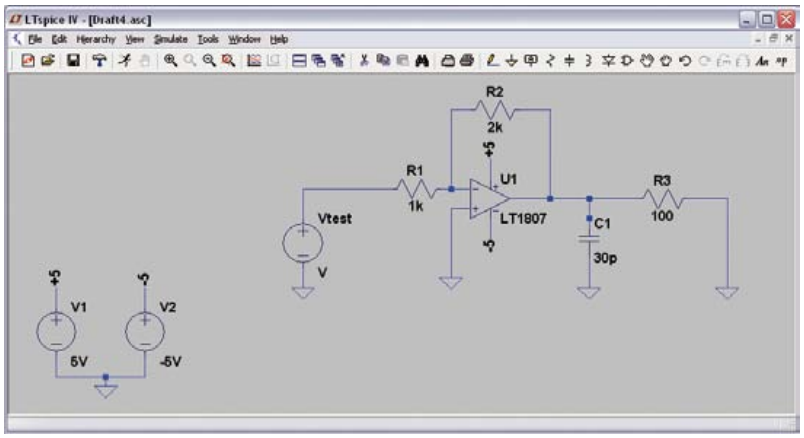
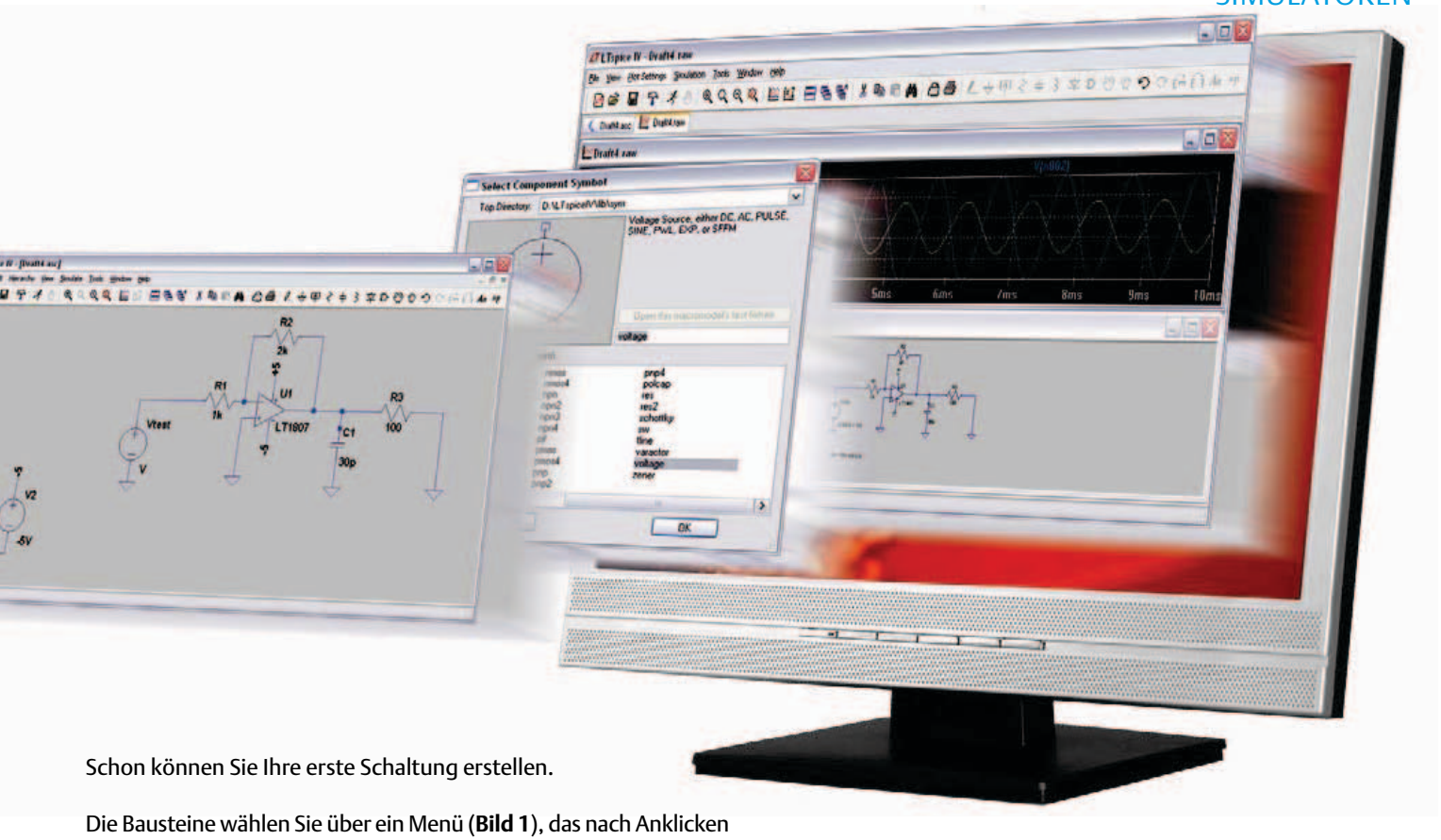


Bild 2. Die erste Schaltung.



Schon können Sie Ihre erste Schaltung erstellen.

Die Bausteine wählen Sie über ein Menü (**Bild 1**), das nach Anklicken der Schaltfläche *Component* erscheint:

Standardkomponenten wie Massepunkte, Widerstände, Kondensatoren, Induktivitäten und Dioden können Sie unmittelbar über die Menüleiste auswählen:

Mithilfe daneben befindlicher Schaltflächen bauen Sie Ihre Schaltung auf:

Der Umgang mit den Komponenten wird schnell deutlich, wenn Sie ein wenig experimentieren. Nützliche Hotkeys sind *Strg R*, um eine Komponente zu drehen, oder *Strg E*, um sie zu spiegeln. In **Bild 2** ist eine kleine Schaltung wiedergegeben, die wir als Beispiel erstellt haben. Bevor Sie diese Schaltung simulieren können, müssen Sie die Werte der Spannungsquelle *Vtest* sowie die Simulationsparameter festlegen. Wenn Sie auf *Vtest* klicken, erscheint ein Menü, in dem Sie die Grundeigenschaften der Spannungsquelle (Gleichspannung und Innenwiderstand) einstellen können. In diesem Fall soll jedoch das

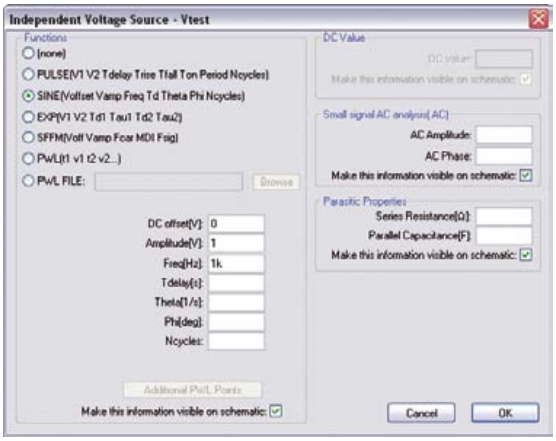


Bild 3. Einstellungen der Signalquelle.

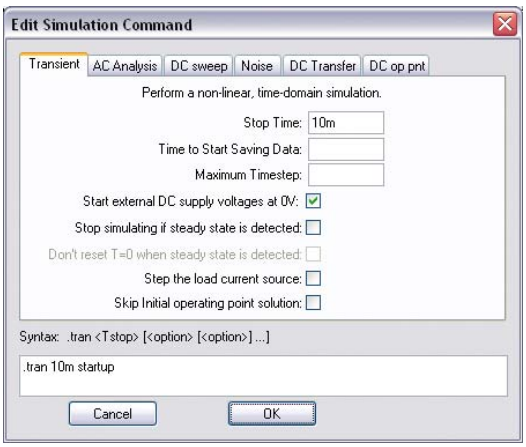


Bild 4. Einstellungen für die Simulation.

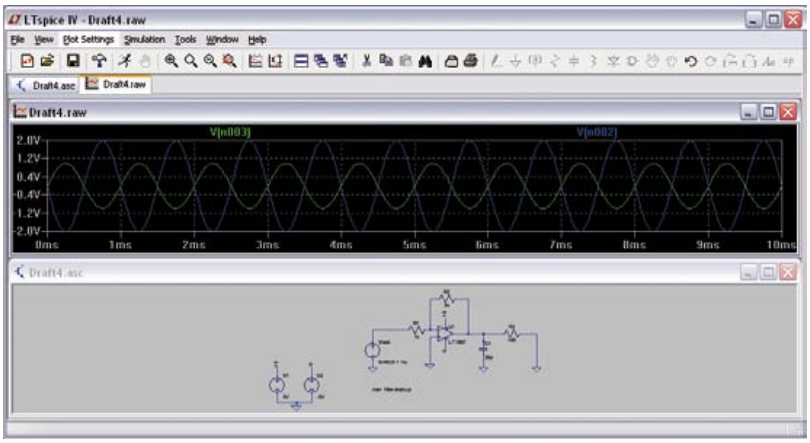


Bild 5. Die simulierten Signale, darunter die Schaltung.

Wechselspannungsverhalten simuliert werden. Zu diesem Zweck führen Sie der Schaltung ein Sinussignal zu (Amplitude 1 V, Frequenz 1 kHz und Offset 0 V). Klicken Sie auf *Advanced*, so dass das in **Bild 3** wiedergegebene Fenster erscheint. Wählen Sie *SINE* und tragen Sie die genannten Werte ein.

Damit die Schaltung im Zeitbereich simuliert werden kann, passen Sie die Simulator-Einstellungen über *Edit Simulation Cmd* an:

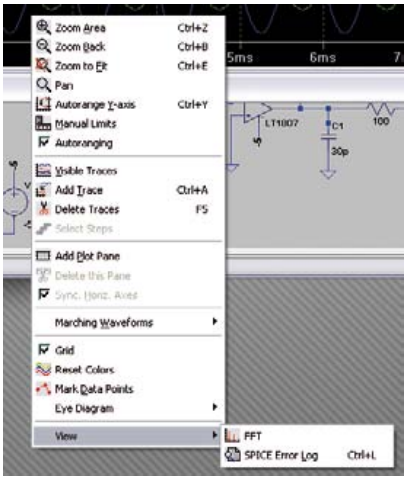


Bild 6. Wahl der FFT-Analyse.

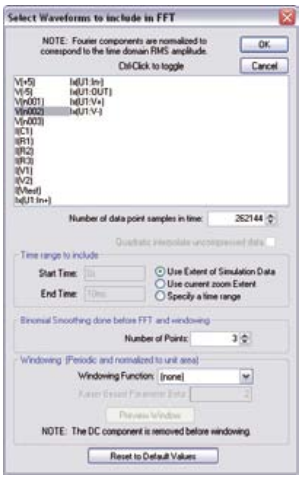


Bild 7. Einstellungen und Signalwahl für die FFT.

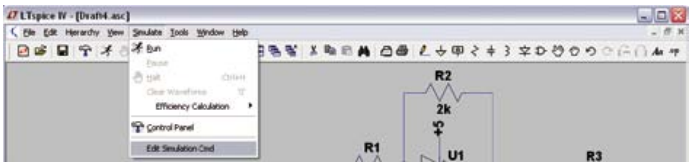
Tipps

- Standard-Komponenten

Beim Erstellen einer Schaltung werden Sie feststellen, dass im Auswahlfenster für bestimmte Komponenten diverse Typ-Varianten erscheinen, während das Fenster anderer Komponenten leer ist. Wenn Sie beispielsweise auf *Opamps* klicken, erhalten Sie eine stattliche Liste, aus der Sie vornehmlich unter Typen von LT wählen können. Dagegen ist bei Wahl eines NPN-Transistors in der Liste kein einziger Typ vermerkt. Eine Typenliste der NPN-Transistoren ist trotzdem vorhanden: Positionieren Sie einen Standard-NPN-Transistor auf dem Arbeitsblatt und klicken Sie mit der rechten Maustaste darauf. Wählen Sie *Pick New Transistor*, so dass eine Liste gängiger Typen und Hersteller erscheint. Dies gilt in gleicher Weise auch für andere Standard-Komponenten.

- Weitere Komponenten

Die Auswahl an Komponenten ist in LTspice naturgemäß begrenzt. Von der Webseite der Yahoo User Group *LTspice*, die Sie unter http://tech.groups.yahoo.com/group/LTspice/ finden, können Sie viele weitere Komponenten herunterladen.



Jetzt erscheint das Fenster, das **Bild 4** zeigt. Weil der Verlauf des Ausgangssignals abhängig von der Zeit betrachtet werden soll, wählen Sie den Tab *Transient*. In diesem Beispiel ist die Dauer der Simulation auf 10 ms eingestellt, die Simulation beginnt bei der Gleichspannung 0 V. Klicken Sie auf *OK* und bewegen Sie das Feld mit den Einstelldaten, das beim Mauszeiger erscheint, auf einen willkürlichen Ort in der Schaltung. Für eine Analyse im Frequenzbereich können Sie den Tab *AC Analysis* wählen und dort die Werte eintragen.
Der Simulation steht nun nichts mehr im Weg: Klicken Sie im Hauptmenü auf das Symbol *RUN*:

Es öffnet sich ein Fenster, das die simulierten Signale darstellt. Wenn Sie nun in der darunter liegenden Schaltung einen Messpunkt anklicken, erscheint das zugehörige Signal in einem Signalfester. Wie **Bild 5** zeigt, wurden in diesem Beispiel der Ausgang der Signalquelle und der Opamp-Ausgang gewählt.
Nachdem Sie rechts auf die Schwingungsform geklickt haben, öffnet sich das in **Bild 6** wiedergegebene Optionsmenü. Dort lassen sich diverse Eigenschaften der Signaldarstellung verändern. Unter *View* können Sie auf *FFT* umschalten (*Fast Fourier Transformation*), die Signale werden dann im Frequenzbereich dargestellt. Im oberen Feld des folgenden Fensters (**Bild 7**) wählen Sie den Messpunkt, auf dessen Signal die FFT angewendet werden soll. Im Beispiel ist dies das Signal am Opamp-Ausgang. Das Ergebnis der FFT ist das im Frequenzbereich dargestellte Signal, zu sehen in **Bild 8**.

Nach diesem Kurzeinstieg in LTspice ist das Entdecken der vielen weiteren Möglichkeiten nicht mehr allzu schwierig. Wir raten Ihnen, auch einen Blick auf andere Simulatoren zu werfen, von vielen Produkten sind Demoversionen oder Versionen für Auszubildende erhältlich. Die Handhabung ist zwar unterschiedlich, grundlegende Vorgänge sind jedoch ähnlich. Dazu gehören das Erstellen der Schaltung, das Einstellen einer oder mehrerer Signalquellen, die Wahl der Parameter für die Simulation und natürlich das Betrachten der Ergebnisse.

(110543)gd

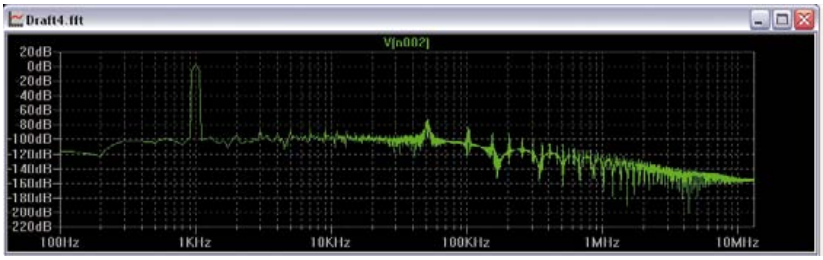


Bild 8. Von der FFT errechnetes Frequenzspektrum.

Exotische Röhren

Von Reginald Neale (USA)

Nach den größeren Beiträgen der letzten beiden Monate mit viel Text und noch mehr Theorie gibt es nun mit dieser Bilderreise durch die vergangene Röhrenepoche etwas leichtere Kost. Erinnern Sie sich noch an die eine oder andere Röhre? Wissen Sie, welche luftleeren Glaskolben bei Ihnen in irgendeiner Schachtel verstauben? An die Röhren-Feinde unter den Elektronikern: Welchen Typ finden Sie besonders gruselig?

Reginald Neale schrieb uns aus Farmington, NY: „In den ersten Jahren meiner Berufstätigkeit als Ingenieur in den frühen 1950ern bestand die Elektronik-Welt ganz aus Vakuum (oder war gelegentlich gasgefüllt). Transistoren wurden noch selten gesichtet. Anbei eine Kollektion eher seltener Röhren-Typen.“

(110387)

Röhren aus den 1930ern wie die eine links hatten 4, 5, 6, oder 7 Anschlüsse in den unterschiedlichsten Abmessungen. Dann kamen der Oktal-Sockel, 9- und 7-polige Miniaturtypen sowie das 12-polige Compactron und Subminiaturtypen.

Eine zweistrahlige Kathodenstrahlröhre von DuMont als Radar-Display aus dem 2. Weltkrieg. Das Sichtfenster hat einen Durchmesser von 5”. Ein Strahl zeigt das Radar-Echo und der zweite schreibt Text dazu.

Eine Eichel-Röhre. Dank planer Konstruktion und radialer Anschlüsse taugte sie auch für das UKW-Band. UHF war in den späten 1940ern die Grenze des Machbaren.

Das Thyratron ist das Röhrenanalogon eines Thyristors. Der Stromfluss wird über ein Steuergitter getriggert und die Röhre bleibt solange leitend, bis der Anodenstrom wieder auf null fällt.

Viele große Namen des Röhrenzeitalters sind heute verschwunden. Wem sagt CBS-Hytron, Philco, Stromberg-Carlson, Capehart, National Union, Admiral, Emerson, Tung-Sol, Sylvania, Muntz, Dumont, Wells-Gardner, Crosley, Raytheon, Motorola oder Zenith noch etwas?

Sende-Empfänger-Röhre eines Radar-Systems aus dem 2. Weltkrieg. Eingebaut in einen Hohlleiter kann sie mittels Hochspannungs-Impulsen sehr schnell zwischen Senden und Empfangen umschalten.

Eine Leistungs-Sende-Tetrode 4X150. Die metallische Anode hat Kühlrippen zur Zwangskühlung. Seinerzeit ein sehr verbreiteter Typ – praktisch jeder Funkamateur hatte so eine Röhre.

Die Victoreen 5841 ist eine Spezial-Subminiatur-Hochspannungs-Regler-Röhre. Sie stellt einen Strom von 5...100 µA bei 900 V zur Verfügung.

In dieser Röhre steckt neben der Heizung ein Thermoelement, mit dem man eine vom Vakuum abhängige Spannung erhält. Oben sieht man den 1/8“-Anschluss.

Diese Röntgen-Röhre aus den 1950ern ist gut 30 cm lang und verwendet eine feststehende Anode und Kathode. Moderne Röntgen-Röhren arbeiten mit rotierenden Anoden.

Miniatur-Fotoröhren waren die Voraussetzung für den Tonfilm der 1930er Jahre. Am Rand des Filmstreifens wurde die Tonspur optisch aufgezeichnet. Spätere Tonfilme nutzten einen Magnetstreifen, der besseren Klang ermöglichte.

Die 832-A ist eine früher bei Funkamateuren beliebte Sende-Doppeltetrode. Sie war im Gegensatz zur europäischen QQE06/40 (siehe Retronik 12/2008) nicht neutralisiert.

Retronik ist eine monatliche Rubrik, die antiker Elektronik und legendärer ELEKTOR-Schaltungen ihre Referenz erweist.
Beiträge, Vorschläge und Anfragen schicken Sie bitte an: editor@elektor.com

Hexadoku

Sudoku für Elektroniker

Nun werden die Tage wieder kürzer und die Nächte länger. Da haben Sie natürlich wieder ein wenig mehr Zeit, sich auf einen bequemen Stuhl zu setzen und mit unserem ganz besonderen Rätsel zu beginnen. Ein wenig Fleiß muss schon sein – aber dann winkt auch ein Preis: Wie immer können Sie einen von vier Gutscheinen gewinnen, wenn Sie uns die richtigen Zahlen in den grauen Kästchen zusenden!

Die Regeln dieses Rätsels sind ganz einfach zu verstehen: Bei einem Hexadoku werden die Hexadezimalzahlen 0 bis F verwendet, was für Elektroniker und Programmierer ja durchaus passend ist.
Füllen Sie das Diagramm mit seinen 16 x 16 Kästchen so aus, dass **alle** Hexadezimalzahlen von 0 bis F (also 0 bis 9 und A bis F) in jeder Reihe, jeder Spalte und in jedem Fach mit 4 x 4 Kästchen (markiert durch die dickeren schwarzen Linien) **genau einmal** vorkommen.
Einige Zahlen sind bereits eingetragen, was die Ausgangssituation des Rätsels bestimmt.
Wer das Rätsel löst - sprich die Zahlen in den grauen Kästchen herausfindet - kann wie jeden Monat einen Hauptpreis oder einen von drei Trostpreisen gewinnen!

Mitmachen und gewinnen!

Unter allen internationalen Einsendern mit der richtigen Lösung verlosen wir
einen **ELEKTOR-Gutschein** im Wert von 100 €
und
drei **ELEKTOR-Gutscheine** im Wert von je 50 €.

Einsenden

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:
Elektor – Redaktion Süsterfeldstr. 25 52072 Aachen
Fax: 0241 / 88 909-77 E-Mail: hexadoku@elektor.de
Als Betreff bitte nur die Ziffern der Lösung angeben!
Einsendeschluss ist der 30. November 2011!

Die Gewinner des Hexadokus aus dem Septemberheft stehen fest!

Die richtige Lösung ist: 4D0F6.
Der Elektor-Gutschein über 100 € geht an: Alf Eriksson aus Åsbro (Schweden).
Einen Elektor-Gutschein über je 50 € haben gewonnen: Monika Häfner, Vincenzo Parisi und M.F. Vidaud.
Herzlichen Glückwunsch!

3					6					2					B
5															6
1	E	6		8			3	C			5		A	0	4
				E			A	8			D				
8	B		5	9		F			C		3	A		2	7
			2	C							6	0			
A		7	3									D	6		C
C	4					5	2	7	9					B	1
E	A	C		F	3					5	0		8	4	D
D				A							4				0
			9		B	8	D	1	2	E		3			
2		3			4	6			A	B			5		F
F		5	0			E			D			6	B		2
	8	A	7		9		B	6		C		5	D	3	
	C			6							8			F	
			E		5	A			3	0		7			

1	9	5	3	8	7	E	A	4	C	F	B	D	2	6	0
C	A	F	6	3	D	9	4	8	0	1	2	E	B	7	5
7	4	D	0	F	6	B	2	5	9	E	A	3	1	C	8
B	8	2	E	5	0	C	1	3	D	6	7	F	4	9	A
E	F	B	9	2	5	D	8	6	3	A	4	1	7	0	C
0	5	A	D	4	3	F	C	1	7	9	E	6	8	B	2
4	1	C	2	6	9	7	E	B	8	0	F	5	A	D	3
6	3	7	8	A	1	0	B	C	2	D	5	9	E	F	4
8	6	E	A	7	F	3	5	D	1	4	C	0	9	2	B
2	B	0	4	9	A	1	D	7	5	3	6	8	C	E	F
9	7	1	5	B	C	8	6	F	E	2	0	4	3	A	D
D	C	3	F	E	4	2	0	9	A	B	8	7	5	1	6
3	D	4	B	C	E	A	7	0	6	5	9	2	F	8	1
5	E	6	7	D	8	4	F	2	B	C	1	A	0	3	9
A	2	9	1	0	B	5	3	E	F	8	D	C	6	4	7
F	0	8	C	1	2	6	9	A	4	7	3	B	D	5	E

Kontakt: Tel. 0511/61 65 95-0 – Fax 0511/61 65 95-55 – E-Mail: service@id-medienservice.de

Starke Stücke

Die ganze Welt der Elektronik in einem Shop!

NEU!

Band 1: Grundlagen

Stromversorgung ohne Stress

Eines haben alle elektronischen Schaltungen und Geräte gemeinsam: ihre Funktion steht und fällt mit der Stromversorgung. Schon deshalb muss man dieser Baugruppe besondere Aufmerksamkeit widmen. Dieses neue Buch beinhaltet Grundlagen und Schaltungen der Stromversorgungstechnik für elektronische Geräte aus der Praxis. Dem aktuellen Trend folgend hat der Autor der mobilen Stromversorgungstechnik und der Schaltnetzteiltechnik besondere Aufmerksamkeit gewidmet. Dabei wird berücksichtigt, dass die Stromkonstanter gegenüber den Spannungskonstantern zunehmend an Bedeutung gewinnen.
Man findet im Buch außer den notwendigen Grundlagen zum Bau eigener Stromversorgungsgeräte und -baugruppen auch ganz praktische Anwendungsbeispiele, etwa für den Ersatz defekter Netztransformatoren, für die mobile Stromversorgung auf Fahrradtouren oder für den Betrieb von LEDs.

294 Seiten (kart. • ISBN 978-3-89576-248-2 • € 38,00 • CHF 41,80

Der 3. Band der neuen Buchreihe

Embedded Electronics 3

Die Bände dieser Reihe wenden sich an jene, die sich – als Auszubildende, Studierende oder Berufseinsteiger – von A bis Z in die professionelle Schaltungs- und Systementwicklung einarbeiten wollen. Sie bieten, was Praktiker und Lernende brauchen: Eine Auffrischung und Vertiefung der Grundlagen, Anregungen zu eigenen Gedanken und Zugänge zu Einzelheiten, Querverbindungen und Spitzfindigkeiten.

412 Seiten (kart.) • ISBN 978-3-89576-185-0
€ 49,00 • CHF 53,90

Von Top-Autor und Entwickler Kainka

Basiskurs BASCOM-AVR

BASCOM und AVR-Controller sind ein starkes Team! Was immer man entwickeln möchte, meist hat ein ATmega schon das Wichtigste an Board: Ports, Timer, AD-Wandler, PWM-Ausgänge und serielle Schnittstelle, RAM, Flash-ROM und EEPROM, alles ist reichlich vorhanden. Und BASCOM macht die Anwendung zu einem Kinderspiel. Auch komplexe Peripherie wie LCD, RC5 und I²C lassen sich mit wenigen Befehlen nutzen.

223 Seiten (kart.) • inkl. Software-CD
ISBN 978-3-89576-238-3 • € 39,80 • CHF 43,80

Bücher

„Die Spannung steigt!“

Stromversorgungen in der Praxis

Die Elektronik bestimmt unser tägliches Leben mehr denn je – Tendenz steigend. Dabei benötigen alle elektronischen Geräte und Systeme eine gut funktionierende Stromversorgung mit spezieller Anpassung an die Betriebsbedingungen. Dieses neue Buch beschreibt die entsprechenden Möglichkeiten vom Transformator bis zum passenden Kühlkörper. Behandelt werden die wichtigsten Merkmale, Einsatzmöglichkeiten und das Betriebsverhalten von vielen unterschiedlichen Stromversorgungsgeräten.

366 Seiten (kart.) • ISBN 978-3-89576-239-0
€ 46,00 • CHF 50,60

MIFARE und kontaktlose Smartcards angewandt

RFID

MIFARE ist die weltweit meistgenutzte RFID-Technologie. Dieses neue Buch bietet einen praxisorientierten und umfassenden Einstieg in diese Technologie. Die einleitenden Kapitel behandeln u. a. die physikalischen Grundlagen, die relevanten Normen, das RFID-Antennendesign, die Sicherheitsaspekte und die Kryptografie. Das vollständige Hardware- und Softwaredesign eines Readers ist ausführlich beschrieben.

464 Seiten (kart.) • ISBN 978-3-89576-219-2
€ 56,00 • CHF 61,60

LCD-Graphik I, verkettete Strukturen I, Zeichenketten, Fädeltechnik I

AVR-Programmierung 3

Dieser dritte Band der Buchreihe zur Assembler-Programmierung von AVR-Mikrocontrollern richtet sich nicht nur an Einsteiger. Auch die C-Programmierer von AVR-Prozessoren profitieren von der Erläuterung der Besonderheiten, die es bei der Assembler-Programmierung zu beachten gilt. Nach der Erläuterung der statischen Datenstrukturen in Buch 2 folgt in diesem Buch der Einstieg in die dynamischen Strukturen. Er beginnt nach einer allgemeinen Einführung mit der einfachsten Struktur, der verketteten Liste. Der letzte Teil führt in den Selbstbau von Fädelsprachen ein, die ein äußerst personalisiertes und projektorientiertes Programmieren erlauben.

319 Seiten (kart.) • ISBN 978-3-89576-231-4
€ 46,00 • CHF 50,60

Weitere Informationen zu unseren Produkten sowie das gesamte Verlagssortiment finden Sie auf der Elektor-Website:

www.elektor.de

Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen
Tel. +49 (0)241 88 909-0
Fax +49 (0)241 88 909-77
E-Mail: bestellung@elektor.de

Mikrocontroller selber programmieren

Das MSP430 Mikrocontroller Buch

Moderne Mikrocontroller werden immer leistungsfähiger und können vielfältige Aufgaben übernehmen, für die vor wenigen Jahren noch ein kompletter Computer nötig gewesen wäre. Dieses Buch eröffnet einen schrittweisen Einstieg in die Welt der Mikrocontrollerprogrammierung und führt mit ausführlichen Anwendungsbeispielen in die Fähigkeiten dieser außergewöhnlichen Prozessorfamilie ein.

296 Seiten (kart.) • ISBN 978-3-89576-236-9
€ 42,00 • CHF 46,20

Visual Studio

C# 2010 Programmierung und PC-Anbindung

Ziel dieses Buches ist, auf einfache Weise zu zeigen, wie mit der populären Hochsprache C# ein PC programmiert werden kann. Am Anfang beschreibt das Buch Datentypen und Programmsteuerungen, die dann um fortschrittliche Konzepte wie die objektorientierte Programmierung, Threads, die Internetkommunikation und Datenbanken erweitert werden. Alle verwendeten Code-Beispiele können kostenlos von der Elektor-Webseite heruntergeladen werden.

349 Seiten (kart.) • ISBN 978-3-89576-244-4
€ 44,00 • CHF 48,40

CD- & DVD-ROMs

Kompletter Elektor-Jahrgang 2010 auf DVD

Elektor-DVD 2010

Die neue Elektor-Jahrgangs-DVD enthält alle Artikel des Jahrgangs 2010. Sie verfügt über eine sehr übersichtlich gestaltete Benutzeroberfläche. Mit der Elektor-DVD 2010 können Sie: Platinenlayouts in perfekter Qualität drucken; diese Layouts mit einem Zeichenprogramm verändern; die Schnellsuchfunktion benutzen, mit der Sie in den einzelnen Artikeln oder im ganzen Jahrgang nach Wörtern, Bauteilen oder Titeln suchen können; Schaltbilder, Platinenlayouts, Illustrationen, Fotos und Texte exportieren.

ISBN 978-90-5381-267-9 • € 27,50 • CHF 30,30

Mikrocontroller-Programmierung leicht gemacht

ATM18-Collection

Diese CD-ROM enthält Artikel der populären Elektor-Serie „CC2-AVR-Projekt“ mit mehr als 25 Projekten mit dem ATM18-Board inkl. der benötigten Software und Platinenlayouts sowie weiteren Zusatzinformationen. Des Weiteren umfasst die CD auch den kompletten 6-teiligen Elektor-BASCOM-AVR-Kurs.

ISBN 978-0-905705-92-7 • € 29,50 • CHF 32,50

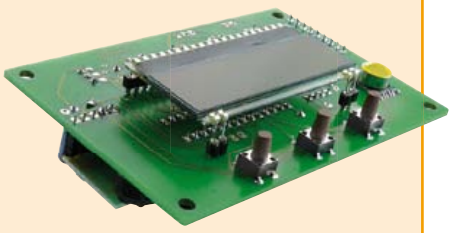
USB-FT232R Breakout-Board

(Elektor September 2011)

Dieser USB-nach-TTL-Wandler ist nicht viel größer als der angegossene Stecker eines USB-Kabels. Seine nützlichen Dienste kann die Mini-Platine unter Windows, Linux und anderen Betriebssystemen entfalten. Mit dem praktischen Konverterboard lassen sich eigene Schaltungen einfach um einen USB-Anschluss erweitern und leicht USB/RS232- oder RS485-Wandler realisieren. Die Platine ist überall dort praktisch, wo TTL erwünscht, aber nur USB vorhanden ist.

Bestückte und getestete Platine

Art.-Nr. 110553-91 • € 15,00 • CHF 16,50



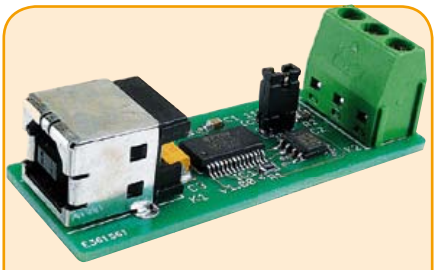
USB-Wetterlogger mit Langzeitspeicher

(Elektor September 2011)

Dieser autonome Datenlogger zeichnet die von I²C-Sensoren gelieferten Daten für Luftdruck, Temperatur und Feuchte auf und zeigt sie auf einem LC-Display an. Die Ergebnisse lassen sich über USB auslesen und mit GNUplot auf einem PC grafisch darstellen. Dank der digitalen Sensormodule ist der Hardwareaufwand gering und ein Abgleich nicht erforderlich. Die Betriebsdauer mit drei Mignonzellen beträgt sechs bis acht Wochen.

Kit bestehend aus Platine, prog. Controller, Feuchte- und Luftdrucksensor

Art.-Nr. 100888-73 • € 34,95 • CHF 38,50



USB/RS485-Konverter

(Elektor Juni 2011)

Der USB/RS485-Konverter stellt die Verbindung zwischen dem USB-Port am PC oder Laptop und RS485-Signalen her. Der Konverter ist speziell für den 2-Draht-RS485-Bus konzipiert, der Anschluss an den Bus erfolgt unkompliziert durch eine dreipolige Klemmleiste. Dank der Verwendung des bekannten FT232L-Schnittstellenwandlers von FTDI stehen virtuelle COM-Port-Treiber nicht nur für alle gängigen Windows-Versionen, sondern auch für Windows CE, Windows Mobile, Linux und Mac OS X zur Verfügung.

Bestückte und getestete Platine

Art.-Nr. 110258-91 • € 24,95 • CHF 27,50

Pico C

(Elektor April 2011)

HF-Entwickler müssen häufig parasitäre Kapazitäten im Picofarad-Bereich aufspüren. Leider werden auch bessere Digitalmultimeter ziemlich unpräzise, wenn es um das Messen solch kleiner Kapazitäten geht. Gewöhnlich liegt der kleinste Messbereich bei 2.000 pF! Mit dem „Pico C“ klappt das Messen besser, er zeigt die Picofarads sogar mit Nachkommastelle an!

Bausatz mit allen Bauteilen inkl. ‘Elektor Project Case’, Platine, progr. Controller und LCD

Art.-Nr. 100823-71 • € 82,50 • CHF 90,80

Bausätze & Module

Lieferprogramm

November 2011 (Nr. 491) €

+++ Das Lieferprogramm zu dieser Ausgabe finden Sie auf www.elektor.de +++

Oktober 2011 (Nr. 490)

AVR-Platine Platino
100892-1 ..... Platine 12,95

Hier kommt der Bus (8)
110258-1 ..... Platine (Experimental-Knoten) 5,95
110258-1C3 ... 3 x Platine (Experimental-Knoten) 12,95
110258-91 ..... USB/RS485-Konverter (bestückt und getestet) 24,95

Audio-DSP-Kurs
110001-91 ..... DSP-Board (bestückt und getestet) 129,95
110001-92 ..... Kit bestehend aus DSP-Board (110001-91) und Programmer (110534-91 / Veröffentlichung im November-Heft / erhältlich ab KW42) 149,95

September 2011 (Nr. 489)

USB-Wetterlogger mit Langzeitspeicher
100888-1 ..... Platine 17,95
100888-41 ..... Programmierter Controller ATmega88-20PU 9,95
100888-71 ..... HH10D Feuchtesensor 7,95
100888-72 ..... HP03SA Luftdrucksensor 6,45
100888-73 ..... Kit bestehend aus Platine, prog. Controller, Feuchte- und Luftdrucksensor 34,95

I²C-Sensoren
100888-71 ..... HH10D Feuchtesensor 7,95
100888-72 ..... HP03SA Luftdrucksensor 6,45

USB-FT232R Breakout-Board
110553-91 ..... Bestückte und getestete Platine 15,00

J²B: Vielseitiges HMI-Modul mit ARM Cortex-M3
050176-74 ..... Gehäuse Bopla Unimas 160 9,95
110274-1 ..... Platine ..... www.elektor.de
110274-71 ..... Kit bestehend aus Platine mit LPC1343-Controller, Quarz, Spannungswandler, LCD- und USB-Interface (bereits bestückt), LED und Stiftleisten ..... www.elektor.de
110274-72 ..... LCD, 4 x 20 Zeichen (HD44780-kompatibel) ... www.elektor.de

Hier kommt der Bus (7)
110258-1 ..... Platine (Experimental-Knoten) 5,95
110258-1C3 ... 3 x Platine (Experimental-Knoten) 12,95
110258-91 ..... Bestückte und getestete Platine 24,95

Twittern mit E-blocks
EB003 ..... E-blocks Sensor-Interface 26,80
EB005 ..... E-blocks LCD-Board 29,75
EB006 ..... E-blocks PIC-Multiprogrammer 89,25
EB007 ..... E-blocks Switch-Board 17,85
EB059 ..... E-blocks Servo-Board 17,85
EB069 ..... E-blocks WLAN-Board 164,95
TEDSSI4 ..... Flowcode 4 für dsPIC (Professional-Version) 221,65

Juli/August 2011 (Nr. 487/488)

Akkutester
110154-41 ..... Progr. Controller PIC16F873A 13,95

Timer für 2-4-6 Stunden
110219-41 ..... Progr. Controller PIC12F675 DIL8 9,95

Morseuhr
110170-41 ..... Progr. Controller ATtiny4520-PU DIP8 9,95

Elex-Experimentierplatine
ELEX-1 ..... Experimentier-Platine Elex-1 5,50
ELEX-2 ..... Experimentier-Platine Elex-2 9,95
ELEX-4 ..... Experimentier-Platine Elex-4 17,95

Baustellenampel für den Modellbau
110203-41 ..... Progr. Controller ATtiny13 DIP8 9,95

Berührungsloses Thermometer
100707-1 ..... Platine 22,95
100707-41 ..... Progr. Controller PIC16F876A DIL28 15,00

R8C/13 spricht CAN / Mehr Portleitungen für den R8C/13
050179-91 ..... R8C/13-Starterkit (aufgebautes Board + Software-CD) .. 13,95

Schweißlicht für die Modellbahn
110085-41 ..... Progr. Controller PIC10F200-I/P DIP8 9,95

Bestseller

Bücher

1 **Embedded Electronics 3**
ISBN 978-3-89576-185-0 ..... € 49,00 ..... CHF 53,90

2 **Stromversorgungen in der Praxis**
ISBN 978-3-89576-239-0 ..... € 46,00 ..... CHF 50,60

3 **AVR-Programmierung 3**
ISBN 978-3-89576-231-4 ..... € 46,00 ..... CHF 50,60

4 **C# 2010 Programmierung** und PC-Anbindung
ISBN 978-3-89576-244-4 ..... € 44,00 ..... CHF 48,40

5 **Basiskurs BASCOM-AVR**
ISBN 978-3-89576-238-3 ..... € 39,80 ..... CHF 43,80

CD- & DVD-ROMs

1 **ECD 6**
ISBN 978-90-5381-258-7 ..... € 29,50 ..... CHF 32,50

2 **ATM18-Collection**
ISBN 978-0-905705-92-7 ..... € 29,50 ..... CHF 32,50

3 **Wireless-Toolbox**
ISBN 978-90-5381-268-6 ..... € 32,50 ..... CHF 35,80

4 **Elektor-DVD 2010**
ISBN 978-90-5381-267-9 ..... € 27,50 ..... CHF 30,30

5 **The Audio Collection 3**
ISBN 978-90-5381-263-1 ..... € 21,50 ..... CHF 23,70

Bausätze & Module

1 **USB-FT232R Breakout-Board**
Art.-Nr. 110553-91 ..... € 15,00 ..... CHF 16,50

2 **USB-Wetterlogger** mit Langzeitspeicher
Art.-Nr. 100888-73 ..... € 34,95 ..... CHF 38,50

3 **USB/RS485-Konverter**
Art.-Nr. 110258-91 ..... € 24,95 ..... CHF 27,50

4 **Pico C**
Art.-Nr. 100823-71 ..... € 82,50 ..... CHF 90,80

5 **OBD-2-Wireless** (Bluetooth)
Art.-Nr. 100872-72 ..... € 124,95 ... CHF 137,50

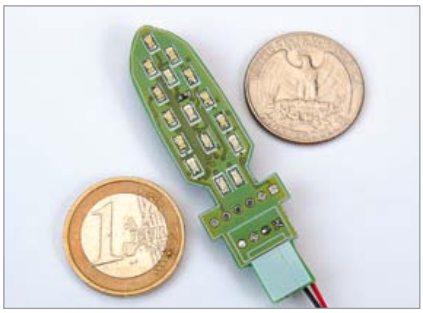
Bestellen Sie jetzt einfach und bequem online unter www.elektor.de/shop oder mit der portofreien Bestellkarte am Heftende!

Elektor-Verlag GmbH
Süsterfeldstr. 25, 52072 Aachen
Tel. +49 (0)241 88 909-0
Fax +49 (0)241 88 909-77
E-Mail: bestellung@elektor.de

Preisänderungen und Irrtümer vorbehalten!

USB-Stick-Datenlogger

Um Daten einer Mikrocontroller-Schaltung über einen längeren Zeitraum aufzuzeichnen, wird oft ein PC angekoppelt. Wenn man eine Möglichkeit findet, einen USB-Stick direkt an das Mikrocontroller-System anzuschließen, geht es auch ohne diese Stromfresser. Mit einem USB-Stick hat man einen bewährten und billigen Speicher, mit dem sich die Daten in idealer Weise transportieren und transferieren lassen. Da der USB-Stick ein USB-Peripheriegerät (Device) darstellt, muss zwischen der Mikrocontroller-Schaltung mit ihrem seriellen Port und dem Stick mit seinem USB-Stecker ein USB-Hostcontroller für die nötige "Konnektivität" sorgen. Eine Aufgabe, die mit einem PIC24FJ64GB002 von Microchip elegant gelöst wurde.



Intelligente LED-Kerze

Wenn es auf Weihnachten zugeht, steigt bei Elektronikern das Interesse an originellem Licht-Baum- und Fensterschmuck – vorzugsweise mit LEDs. In der Pionierzeit von Elektor gab es schon einmal eine elektronische Wunderkerze mit einem Glühbirnchen, das sich wie der Docht einer traditionellen Wachskerze „anzünden" und „auspusten" ließ.
Die aktualisierte Version ist im 21. Jahrhundert – wie könnte es anders sein – mit SMD-LEDs und einem Mikrocontroller (PIC 16F1827) bestückt.

Interface für Breitband-Lambdasonde

Für die Analyse eines Verbrennungsvorgangs – beim Auto wie bei der Zentralheizung – kann man mit einer Breitband-Lambda-Sonde den Sauerstoffgehalt im Abgas messen. Allerdings ist die Ansteuerung einer Breitband-Lambdasonde nicht ganz einfach.
Die Schaltung verwendet daher eine Breitband-Lambdasonde von Bosch mit einem speziell dafür entwickelten IC des gleichen Herstellers. Mit dieser Kombination ist nicht nur eine genaue Messung möglich - die Schaltung kommt sogar ohne Abgleich aus. Am Ausgang kann man einfach ein TTL-UART-Interface anschließen.

Elektor Dezember erscheint am 16. November 2011.
Elektor gibt es im Bahnhofsbuchhandel, Elektronik-Fachhandel, an ausgewählten Kiosken und garantiert beim Presse-Fachhändler.
Sie können Elektor auch direkt bei www.elektor.de bestellen.

Änderungen vorbehalten!

Abo-Service:
E-Mail: abo@elektor.de

Bestellannahme und Bestellservice:
E-Mail: bestellung@elektor.de

Geschäftszeiten
Montag – Donnerstag von 08:30 bis 17:00 Uhr
Freitag von 08:30 bis 12:30 Uhr
Tel. +49 241 88 909-0
Fax +49 241 88 909-77

Unser Kundenservice berät Sie bei allen Fragen zu Bestellungen, Lieferterminen und Abonnements. Änderungen, Reklamationen oder besondere Wünsche (wie z. B. Geschenkabonnement) richten Sie ebenfalls an den Kundenservice. Vergessen Sie bitte nicht, Ihre Kundennummer anzugeben – falls vorhanden.

Einzelheft

Deutschland	€ 7,40
Österreich, Belgien, Luxemburg	€ 8,20
Schweiz	CHF 14,30

Jahresabonnement-Standard

Deutschland	€ 77,50
Österreich, Belgien, Luxemburg	€ 84,50
Schweiz	€ 97,50
Andere Länder	€ 102,50

Jahresabonnement-PLUS

Deutschland	€ 95,00
Österreich, Belgien, Luxemburg	€ 102,00
Schweiz	€ 115,00
Andere Länder	€ 120,00

Probeabonnement

Alle Länder (zzgl. Porto)	€ 14,90

Studentenabo-Standard

Deutschland	€ 62,00
Österreich	€ 67,60
Schweiz	€ 78,00

Studentenabo-PLUS

Deutschland	€ 79,50
Österreich	€ 85,10
Schweiz	€ 95,50

Upgrade zum Abo-PLUS

Alle Länder	€ 17,50

Jahres- und Studentenabonnements (11 Hefte) dauern immer 1 Jahr und verlängern sich automatisch um weitere 12 Monate, wenn nicht spätestens 2 Monate vor Ablauf schriftlich gekündigt wird.

Änderungen und Irrtümer vorbehalten.

Bankverbindungen
Commerzbank Aachen
Konto 1 201 102 (BLZ 390 400 13)
IBAN: DE89 3904 0013 0120 1102 00
BIC: COBADEFFXXX

Postgiro Köln
Konto 229 744-507 (BLZ 370 100 50)
IBAN: DE17 3701 0050 0229 7445 07
BIC: PBNKDEFF

Elektor-Bestellkarte 11/2011

Ja, ich möchte Elektor im Jahresabonnement

(11 Hefte / inkl. Doppelheft Juli/August) pünktlich und zuverlässig frei Haus beziehen*.
Im Vergleich zum Einzelheftkauf am Kiosk spare ich beim Standard-Abonnement € 10,00 (bei der PLUS-Variante sogar € 25,00). Als Dankeschön erhalte ich das Elektor-Buch "Software Defined Radio" (sofort nach Zahlung der Abonnementsrechnung) gratis zugeschickt.

Bitte wählen Sie Ihr Jahresabonnement aus:

☐ **Jahresabonnement-Standard für nur € 77,50**

☐ **Jahresabonnement-PLUS (inkl. Jahrgangs-DVD 2011** + exklusiver Online-Zugang zu Elektor-Plus.de) für nur € 95,00**

TIPP

Zahlungsweise ☐ Rechnung ☐ Bankeinzug (gilt nur für D)

Bank

Konto

BLZ

Datum, Unterschrift

*Das Abonnement verlängert sich automatisch um 12 Monate, wenn nicht spätestens zwei Monate vor Ablauf schriftlich gekündigt wird.
**Diese DVD-ROM wird Ihnen sofort nach Erscheinen (Februar 2012) zugeschickt.

11/2011

Ja, ich möchte Elektor kennenlernen!

Ich erhalte die nächsten 3 Ausgaben für nur € 14,90 pünktlich und zuverlässig frei Haus.*

Wenn Sie innerhalb von 1 Woche nach Erhalt der dritten Ausgabe nichts von mir hören, möchte ich Elektor im Jahresabonnement für nur € 77,50 weiter beziehen.

Zahlungsweise ☐ Rechnung ☐ Bankeinzug (gilt nur für D)

Bank

Konto

BLZ

Datum, Unterschrift

*Dieses Angebot gilt nur, wenn Sie während der letzten 12 Monate noch nicht Abonnent waren.

11/2011

Diesen Streifen an den unten stehenden Streifen kleben!

Ich bestelle folgende Elektor-Produkte:

Bezeichnung	Preis	Anzahl	Gesamtpreis
Stromversorgung ohne Stress 1 NEU	€ 38,00		
Embedded Electronics 3 NEU	€ 49,00		
AVR-Programmierung 3	€ 46,00		
Stromversorgungen in der Praxis	€ 46,00		
C# 2010 Programmierung und PC-Anbindung	€ 44,00		
Basiskurs BASCOM-AVR	€ 39,80		
ECD 6	€ 29,50		
Elektor-DVD 2010	€ 27,50		
Elektor-Gesamtkatalog			X

zzgl. Porto- und Versandkosten € 5,00

Gesamtbetrag €

Datum: **Unterschrift:**

Tragen Sie bitte Ihre Anschrift auf der Rückseite ein!

Diesen Streifen an den oberen Streifen kleben!

Elektor-PCB-Service

Die Adresse für Platinen, Prototypen und Multilayer

Möchten Sie Ihre selbst entworfene Platine schnell und zuverlässig geliefert bekommen? In Kleinserie und dabei zu einem unschlagbar günstigen Preis? Bestellen Sie jetzt Ihre individuelle Platine beim Elektor-PCB-Service!

elektor PCB Service

Elektor-PCB-Service ist der Leiterplatten-Service von Elektor! Über die Website **www.elektorpcbservice.de** können Sie Ihren Entwurf als professionelle Platine herstellen lassen. Der Elektor-PCB-Service ist die richtige Adresse für Prototypen von neu entwickelten Platinen und für die Produktion modifizierter Elektor-Platinen.

Brauchen Sie kurzfristig einige Muster (Protos) oder eine Kleinserie (Batch), bestehend aus 5 bis 50 Exemplaren? Der Elektor-PCB-Service bietet jetzt beides zu einem günstigen Preis. Sie müssen uns nur über unsere Website Ihr Platinenlayout zusenden.

- Höchste Präzision und Industrie-Qualität zum günstigen Preis
- Kein Mindestbestellwert
- Keine Film- oder Einrichtungskosten
- Keine versteckten Kosten
- Online-Preisrechner
- Versand innerhalb von 5 Werktagen

Überzeugen Sie sich selbst vom Elektor-Leiterplatten-Service – jetzt unter

www.elektorpcbservice.de!

Hier ist meine Anschrift:

Firma

Vorname

Name

Straße, Nr.

PLZ, Ort

Land

DE

Kunden-Nr.

E-Mail

Innerhalb Deutschlands kein Porto nötig!

Antwort

Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen

Hier ist meine Anschrift:

Firma

Vorname

Name

Straße, Nr.

PLZ, Ort

Land

DE

Kunden-Nr.

E-Mail

Innerhalb Deutschlands kein Porto nötig!

Antwort

Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen

Hier ist meine Anschrift:

Firma

Vorname

Name

Straße, Nr.

PLZ, Ort

Land

DE

Kunden-Nr.

E-Mail

Innerhalb Deutschlands kein Porto nötig!

Antwort

Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen

Elektor Electronic Toolbox

Erhältlich im App Store

Umfangreiche Elektroniker-App von Elektronikern für Elektroniker

Die neue „Elektor Electronic Toolbox"-App ist ganz auf die Belange von Elektronikern zugeschnitten. 29 Einzelprogramme/Anwendungen können über eine übersichtliche Oberfläche ausgewählt werden.
Sehr hilfreich im Entwickleralltag sind die Datenbanken für die Bauteilgruppen Bipolar-Transistoren, FETs, Triacs, Thyristoren, Dioden und ICs. Ein Bauteil kann anhand der Typenbezeichnung kinderleicht ausgewählt werden – eine Internetverbindung ist nicht notwendig. Insgesamt sind über 45.000 Bauteile in den Datenbanken verzeichnet. Hinzu kommt eine Spezialdatenbank, in der die Belegung einer Vielzahl von Steckverbindern aus den Bereichen Audio & Video, Computertechnik und Telefon nachgeschlagen werden kann. Nützlich sind auch die interaktiven Bauteilwert-Kalkulatoren.
Tools wie eine virtuelle Widerstandsuhr, ein Umrechner zwischen Maßeinheiten, eine Schaltsymboldatenbank und vieles mehr runden die Elektor-App ab.
Die neue „Elektor Electronic Toolbox" (geeignet für iPhone, iPod und iPad) kann zum Preis von nur 4,99 Euro heruntergeladen werden.

INSERENTENVERZEICHNIS NOVEMBER 2011

Elektor OSPV

Wheelie reloaded:
Das Open-Source-Projekt auf 2 Rädern

Mit unserem beliebten ElektorWheelie (Elektor 06/2009) sind inzwischen viele Menschen unterwegs, die Technik des balancierenden Fortbewegungsmittels fasziniert Jung und Alt. Hinter dem neuen Open Source Personal Vehicle (kurz „OSPV") stehen die gleichen Ideen – Unterschiede gibt es jedoch in der Umsetzung. Der leichter gebaute, zusammenlegbare „OSPV" wurde für den Indoor-Betrieb entwickelt, Konstruktion und Software sind wiedermal Open Source. Auch mit diesem innovativen Einachser ist maximaler Fahrspaß garantiert!

Technische Daten:

- 2 x 200-W-DC-Getriebemotoren
- 2 x 12-V-Bleigel-Akkus mit 9 Ah
- 2 x PU-Räder mit 14 cm Durchmesser
- Zahnriemenantrieb
- Höchstgeschwindigkeit: 15 km/h
- Reichweite: ca. 8 km
- Gewicht: 25 kg
- Maße (H x B x T): 120 x 47 x 47 cm
- Belastung: 90 kg
- Bodenfreiheit Trittflächen: 2 cm
- Höhe Trittflächen: 5,6 cm
- Breite zwischen Trittflächen: 29,5 cm
- Ladezeit: 2,5 Stunden

Der „Elektor OSPV"-Komplettbausatz umfasst zwei DC-Motoren (je 200 W), zwei 12-V-Bleigel-AGM-Akkus, zwei PU-Räder mit Riemenscheibe und HTD-Zahnriemen, Gehäusesatz inkl. aller Schrauben, bestückte und getestete Controller-Platine mit aufgesteckter Sensor-Platine + Ladegerät.

Art.-Nr.: 110320-91
Preis: 1095,00 € (inkl. Mwst., zzgl. Porto/Versand)

elektor

Weitere Infos und Bestellung unter
www.elektor.de/ospv

Elektronikring 1, 26452 Sande
Telefon: +49 (0)4422 955-333
Telefax: +49 (0)4422 955-111

Anzeigenforum: www.explore.reichelt.de
Tagespreise! - Preisstand: 29. 9. 2011

reichelt.de elektronik

Antec

CPU-Kühler Antec H_2O 620

Der spannungsgesteuerte 120-mm-Lüfter und die Kupferkühlplatte der 3. Generation garantieren eine effiziente Kühlung.

- Socket: 775, 1156, AM2, AM2+, 1366, AM3, AM3+, 1155/1156

ANTEC H2O 620
55,95

ARCTIC

CPU-Kühler FREEZER 7 PRO REV.2

- Intel Socket 1366, 1156, 775
- AMD Socket AM3, AM2+, AM2 und 939
- Ausgezeichnete Kühlleistung – 130 Watt
- flüsterleiser 92m PWM-Lüfter
- 6 Heatpipes und 42 Finnen zur effizienten Kühlung
- ARCTIC MX-2 Wärmeleitpaste voraufgetragen
- Lautstärke: 0,8 Sone

AC FREEZER 7PRO2
18,15

Katalog kostenlos!
Tagesaktuelle Preise:
www.reichelt.de

Markenqualität - Top-Service - günstige Preise!

ebm papst Lüfter

ebm-papst Lüfter sind seit Jahrzehnten das Maß der Dinge in der Elektronikkühlung. Kompakt, leise und hocheffizient passen die Energiesparlüfter sich der Kühlsituation an und lassen sich intelligent mit der Gerätelogik vernetzen.

Axiallüfter eignen sich für hohe Luftleistungen bei mittlerem Druckaufbau.

ebmpapst

AC-Axial-Lüfter, 230 V

mit Gleitlager		Maße (mm)	W	U/Min	dB	m³/h
PAPST 8550 N	19,90	80 x 80 x 38	10,5	2150	30	50
PAPST 4550 N	20,75	119 x 119 x 38	15	2550	44	145
PAPST 4580 N	20,95	119 x 119 x 38	18	2350	41	123
PAPST 4650 N	17,95	119 x 119 x 38	18	2650	46	160
PAPST 4890 N	22,95	119 x 119 x 38	10	1550	23	80
mit Kugellager						
PAPST 8556 N	24,95	80 x 80 x 38	10	2800	31	50
PAPST 8556 VW	28,95	80 x 80 x 38	12	2650	35	45
PAPST 3956	21,90	92 x 92 x 25	11	2650	35	59
PAPST 9956	18,50	119 x 119 x 25,4	14	2450	41	117
PAPST 4656 N	22,95	119 x 119 x 38	18	2650	47	160
PAPST 4656 ZW	24,90	119 x 119 x 38	19	2700	44	160

ebmpapst

AC-Axial-Lüfter, 230 V, rund

- Wechselspannungslüfter mit Außenläufer-Kondensatormotor
- geschützt gegen Überlastung durch integr. Thermoschalter
- Lüftergehäuse und Lüfterflügelrad aus Metall

		Maße (Ø/mm)	W	U/Min	dB	m³/h
PAPST 7450 ES	48,50	150 x 172 x 55	47	2700	58	390
PAPST 7855 ES	42,95	150 x 172 x 55	29	2800	49	325
PAPST 7856 ES	49,95	150 x 172 x 55	24	2800	49	325

ebmpapst

DC-Axial-Lüfter, 12 V

mit Gleitlager		Maße (mm)	W	U/Min	dB	m³/h
PAPST 412 F	10,95	40 x 40 x 10	0,7	5400	26	8
PAPST 412 FH	15,35	40 x 40 x 10	0,8	6000	29	9
PAPST 412	12,95	40 x 40 x 20	0,9	6000	18	10
PAPST 612 NGLE	17,75	60 x 60 x 25	0,6	2500	16	21
PAPST 612 NGN	14,95	60 x 60 x 25	1,6	5100	35	42
PAPST 8412 NGLE	16,65	80 x 80 x 25	0,45	1500	12	33
PAPST 3412 N-2G	17,50	92 x 92 x 25	2,2	2700	32	84
PAPST 3412 NG	17,60	92 x 92 x 25	2,2	2700	32	84
PAPST 3412 NGMV	20,95	92 x 92 x 25	1,5-2,0	1400/2300	14-28	44-72
PAPST 4412 F2GML	18,95	119 x 119 x 25	2,0	1950	32	114
PAPST 4212 NGN	23,95	119 x 119 x 38	4,3	2750	42	165
mit Gleitlager + Drehzahlüberwachung						
PAPST 412 F 2H	13,95	40 x 40 x 10	0,8	5400	22	8

ebmpapst

DC-Axial-Lüfter, 24 V

mit Gleitlager		Maße (mm)	W	U/Min	dB	m³/h
PAPST 414 F	10,85	40 x 40 x 10	0,7	5400	26	8
PAPST 414	12,40	40 x 40 x 20	1,0	6000	18	10
PAPST 614 NGN	14,95	60 x 60 x 25	1,8	5100	35	42
PAPST 8414 NGL	16,70	80 x 80 x 25	0,7	1500	12	33
PAPST 8414 NGM	16,70	80 x 80 x 25	1,4	2600	26	58
PAPST 3414 NG	17,60	92 x 92 x 25	2,3	2700	32	84
PAPST 4184 NGX	34,50	119 x 119 x 38	3,5	2800	44	160
PAPST 4214 NGN	23,95	119 x 119 x 38	4,3	2750	42	165
mit Kugellager						
PAPST 414 JHH	25,40	40 x 40 x 25	3,6	13000	46	24
PAPST 3314	19,20	92 x 92 x 32	2,6	3000	37	80
PAPST 4314	20,95	119 x 119 x 32	5,0	2800	45	170
PAPST 4184 NX	34,50	119 x 119 x 38	4,5	3200	49	180