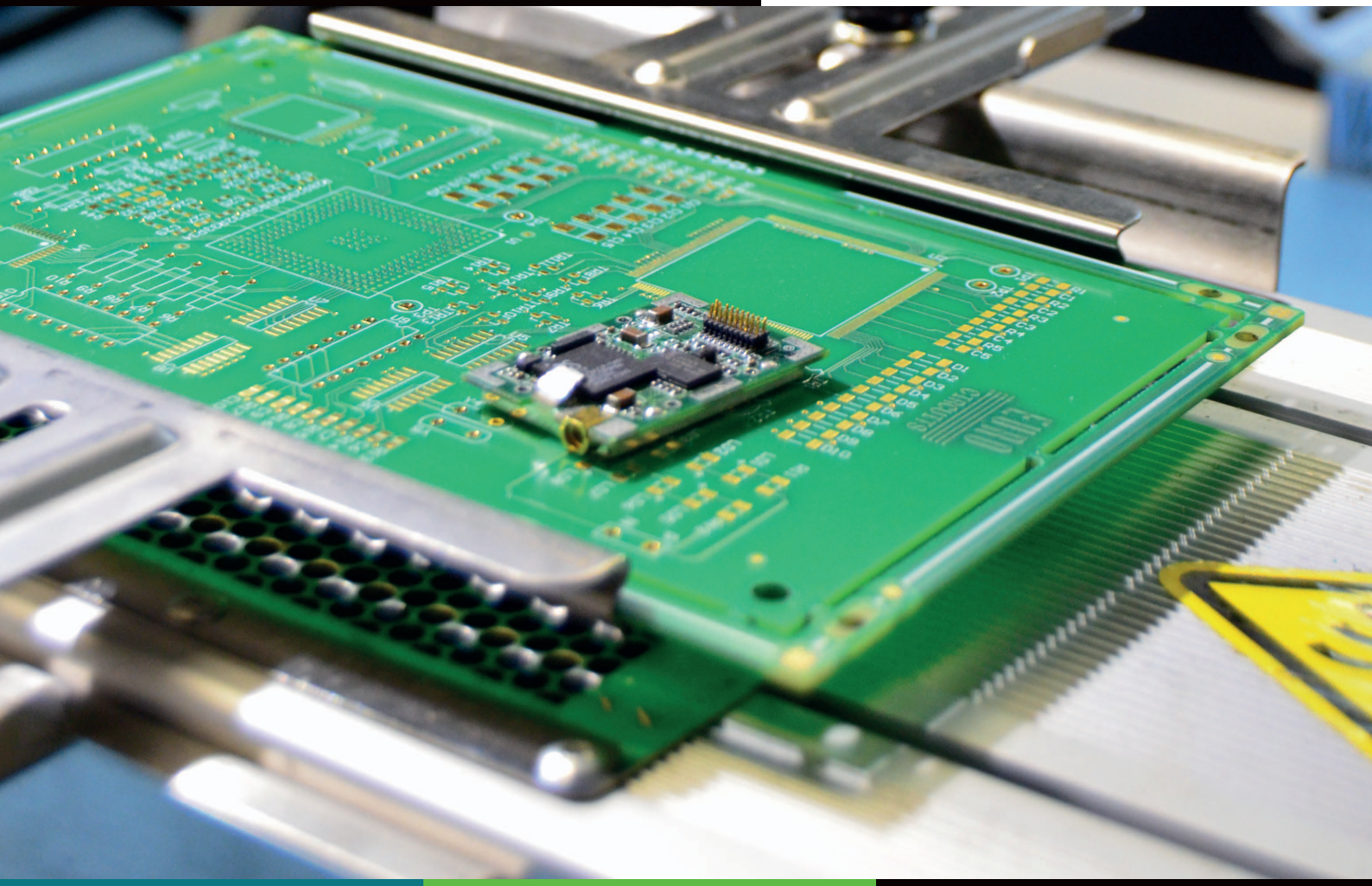
embedded projects
JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

Eine Open-Source Zeitschrift
zum Mitmachen!

PROTOTYPING

[PROJECTS]

- Workshops in Augsburg
- PWM mit FPGA
- Ethernet für STM32F4 Discovery Board
- USB-CAN Adapter
- Vom Überweisungsauftrag zur TAN
- Distance Measurement



Einleitung

Ausgabe 04/2012

Embedded Projects Journal - Ausgabe No. 15

Einleitung

Wer rastet der rostet. Naja so alt sind wir jetzt auch nicht, aber Bewegung ist das, was allen gut tut. Wir schauen mit einem positiven Blick in das vergangene und ebenfalls mit einem positiven Blick in das neue Jahr. Viel hatten wir uns vorgenommen und auch einiges umgesetzt.

Aktuell freuen wir uns neue Sponsoren im Journal begrüßen zu dürfen. So haben wir Eurocircuits[1] als weiteren DIN A4 Sponsor im Boot. Und damit sich so ein Boot stabil im Wasser bewegt, müssen viele Positionen gut besetzt sein. Neben den zahlreichen freiwilligen Autoren, Lesern und Sponsoren gilt an dieser Stelle auch ein besonderer Dank dem Team rund ums Journal. Regelmäßig entsteht mit viel Arbeit und Fleiß eine komplette Ausgabe. Das sind immer 1 – 2 Wochen intensive Arbeit, bis so ein Heft in Form gegossen ist.

Kurz sei an dieser Stelle noch auf den Artikelwettbewerb[2] gemeinsam mit www.mikrocontroller.net hingewiesen. Mikrocontroller.net veranstaltet mit dem embedded projects Journal zum zweiten Mal einen Artikelwettbewerb: Schreibe bis zum 1.3.2013 einen Artikel (im Artikelbreich bei mikrocontroller.net) über ein Elektronik- oder Embedded Systems-Thema, um am Wettbewerb teilzunehmen. Die besten Artikel werden prämiert und im embedded projects Journal veröffentlicht.

In diesem Sinne, Ahoi! Leinen los und auf nach 2013!

Benedikt Sauter

und das embedded projects Team!

[1] http://www.eurocircuits.com

[2] http://www.mikrocontroller.net

Hinweis: Zu jeden Artikel gibt es wieder einen Forumsbeitrag zur Diskussion

http://journal.embedded-projects.net/forum.php (Weiterleitung zu Mikrocontroller.net)

→shop.embedded-projects.net

HARDWARE FOR YOUR PROJECTS – ONLINESHOP

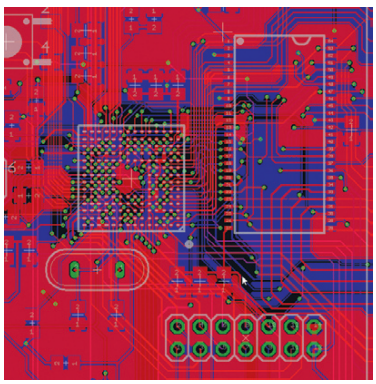
Design your GNUBLIN

Sie suchen ein Board, das fast so wie GNUBLIN ist und brauchen davon nur eine kleine Menge pro Jahr? Wir passen Ihnen auf kurzem Weg die Schaltung an und ergänzen diese nach Ihrem Wunsch. Dank unserer internen Bestückung können wir Ihnen bereits ab kleinen Mengen ähnliche Stückpreise wie bei den GNUBLIN Boards in diesem Shop liefern.

→ **www.gnublin.org/designer**

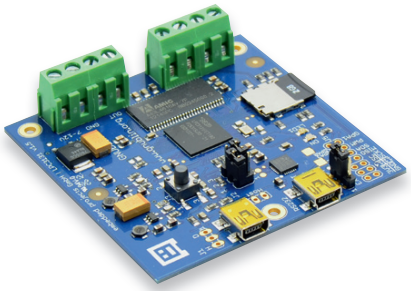
Die einfachste Möglichkeit an ein eigenes embedded GNU/Linux Board zu kommen!
Einfach und genial ist das Produkt „Design your GNUBLIN“. Für alle, die kleine Serien von Platinen mit embedded GNU/Linux benötigen. Basierend auf der Schaltung der Boards der GNUBLIN Familie können wir einfach und schnell kundenspezifische Lösungen erstellen. Innerhalb kürzester Zeit können wir Ihnen die Boards von 1 bis ca. 1000 Stück liefern.

NEU: online GNUBLIN-Designer mit Kalkulator



Wir zeichnen den Schaltplan und das Layout ...

... bestücken mit einem Automaten in Augsburg ...



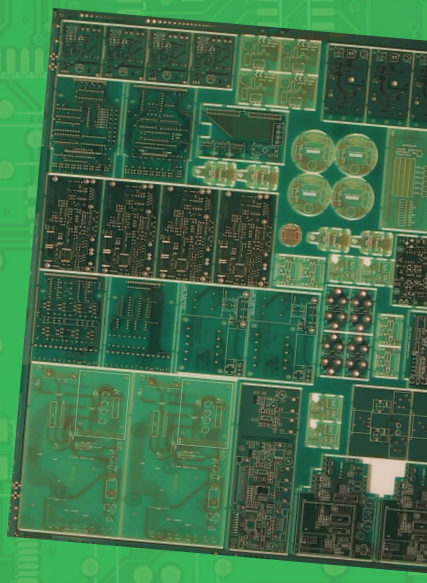
... und nehmen die Schaltung für Sie in Betrieb.

Holzbachstraße 4, D-86152 Augsburg
Tel +49 (0) 821 279599-0
Fax +49 (0) 821 279599-20
shop@embedded-projects.net

embedded projects GmbH
HARDWARE FOR PROJECTS

EURO CIRCUITS

Die Europäische Referenz für PCB Prototypen und Kleinserien



Kostensenkung durch Online-Pooling

- Keine Einrichtkosten
- Keine Mindesbestellwerte - ab der 1. Platine
- Sofortbestellung Online - ohne Vorkasse

Zeitgewinn durch Online-Daten-Check

- PCB Visualizer® - Sofort Design-Rule-Check
- Online Tipps für DFM (Design-For-Manufacturability)
- Online-Optimierung mit Galvano-Simulation

PCB proto – *spezieller Prototypen-Service für Entwickler, preiswert und schnell*

- 1 oder 2 LP in 2, 3, 5 oder 7 Arbeitstagen
- DRC-geprüft, professionelle Ausführung inkl. 2x Lötstopplack und 1x Bestückungsdruck, 150µm Technologie
- 1 x 100 x 80mm in 7AT - 2 Lagen 46.26 € - 4 Lagen 93.94 €
- 2 x 100 x 80mm in 7AT - 2 Lagen 36.28 € je LP - 4 Lagen 73.52 € je LP

Preise inkl. 19% MwSt und ohne Transportkosten

STANDARD pool – *die größte Auswahl an Eurocircuits Pooling Optionen*

- 1-8 Lagen 150µm Technologie-Leiterplatten
- ab 2 AT

TECH pool – *100µm-Technologie mit allen Pooling-Vorteilen*

- 2-8 Lagen 100µm Technologie-Leiterplatten
- ab 4 AT

IMS pool – *Aluminiumkern-Leiterplatten für hohe Wärmeableitung (z.B. LED-Anwendung)*

- Leiterplatten mit einlagig isoliertem Metallsubstrat
- ab 3 AT

On demand – *Alle Optionen im Nicht-Pooling für Spezialanwendungen*

- 1-16 Lagen bis 90µm-Technologie
- ab 2 AT

www.eurocircuits.de

Workshops in Augsburg

Benedikt Sauter <sauter@embedded-projects.net>

Wer kennt diesen Satz nicht: „Das wollte ich schon lange mal gemacht haben...“

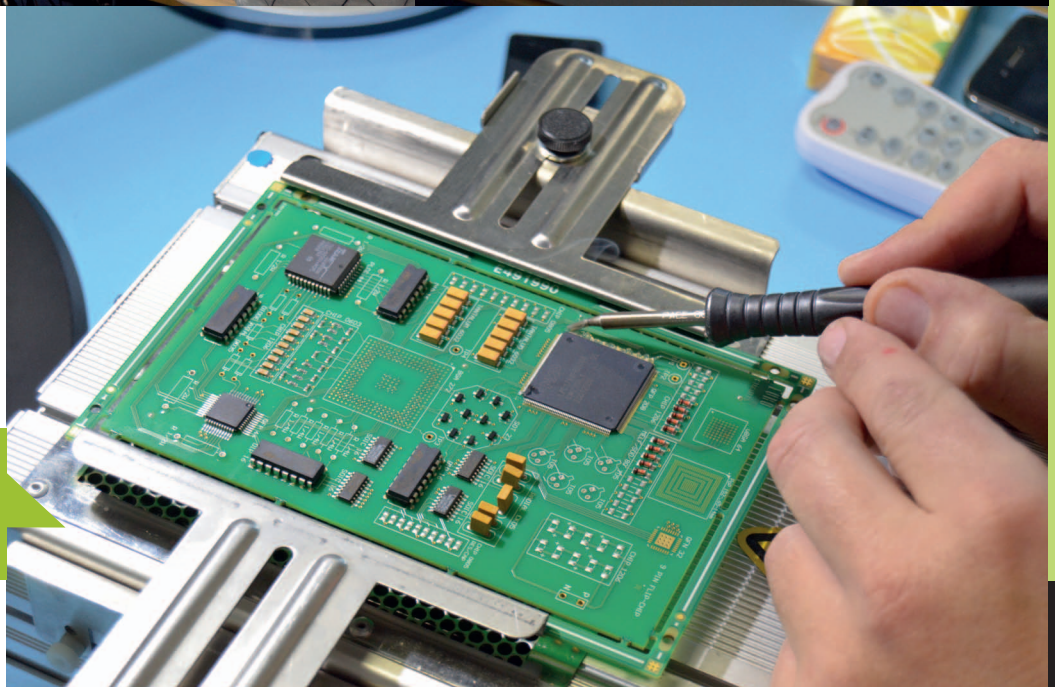
... aber es ist schließlich nicht ganz so einfach, verschiedene Projekte im Alltag oder in der Freizeit unterzubringen. Ganz in diesem Sinne präsentiert unser Journal regelmäßig neue Themen und verschafft so den Einblick auf den Schreibtisch anderer Entwickler. Neu zum Journal kamen Ende 2012 zum ersten Mal auch Workshops bei uns in Augsburg hinzu.

Reflow-Ofen, Schablonendrucker, Entlötgerät, ...

Uwe Dörr (li.), Benedikt Sauter (re.) und Ben Verwaest (hinten)

Die Idee hatten wir schon länger und immer wieder mal in unseren Köpfen, nur gab es nie den richtigen Anstoß. Diesen hat uns schließlich die Firma Eurocircuits [1] aus Belgien geben. „Habt Ihr nicht Lust, gemeinsam mit uns einen Workshop über das Bestücken von SMD-Platinen anzubieten?“

An dieser Stelle ein großer Dank an Uwe Dörr :)
Es hat richtig Spaß gemacht!



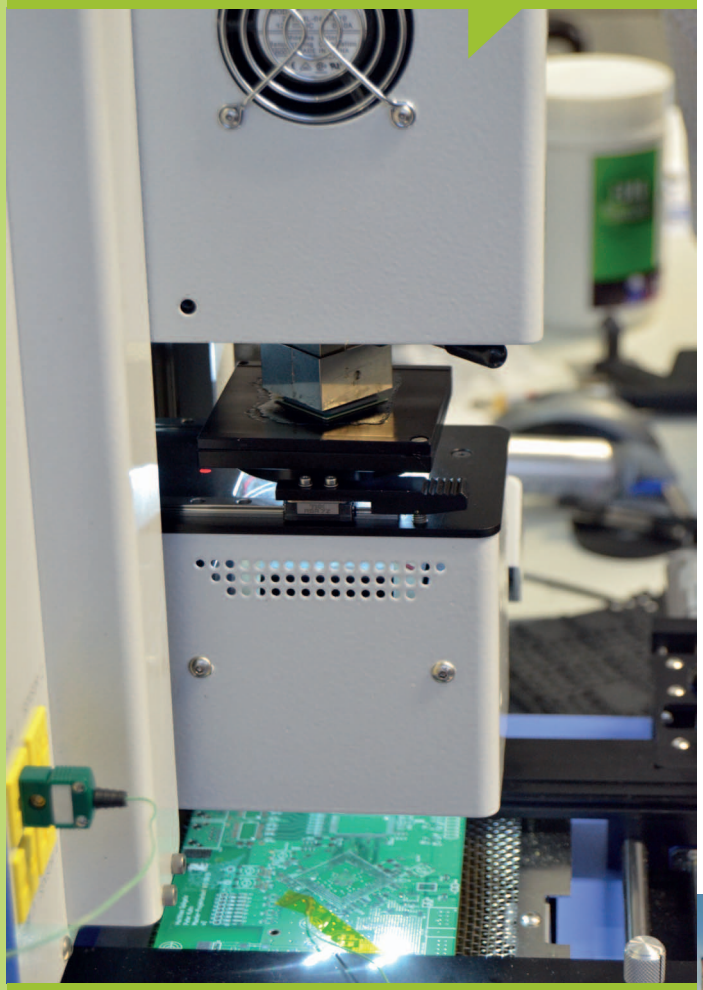
Löten der Mikrocontroller mit Flux, die Platine wurde zuvor mit Paste bedruckt, von Hand bestückt und im Ofen gelötet

Was braucht man um einen Workshop anbieten zu können? Einen passenden Raum, genügend Tische und Stühle, zwischendrin etwas zum Essen für die Bäuche und natürlich genügend Kaffee und Getränke – alles machbar.

Ben Verwaest in der ersten Kaffeepause

Die ersten beiden Workshops gemeinsam mit Eurocircuits waren ein voller Erfolg. Aufgrund der vielen Anmeldungen für den Workshop SMD-Bestückung mussten wir gleich einen zweiten Termin am Tag darauf öffnen; sonst wäre alles aus den Nähten geplatzt. Ebenfalls war der Eagle-Kurs so schnell voll, wie wir alle gar nicht schauen konnten.

BGA Lötmaschine



Die Idee: aus den verschiedenen Disziplinen - von der Hardware, E-Technik bis zur Informatik - regelmäßig kostenlose Workshops anzubieten. Los ging es daher mit den folgenden Themen:

- SMD Bestückung von Platinen
- Die Neuerungen bei Eagle 6.0 / Bibliotheken anlegen mit Eagle

Wir sind also bereit für die nächsten Termine:

- 06.02.2013 Arbeiten mit dem Platinen-Layout Programm Eagle (Richard Hammerl)
- 13.03.2013 embedded GNU/Linux als Mikrocontroller-Ersatz (Benedikt Sauter)

In Zukunft wollen wir zu den eurocircuits Workshops auch eigene bzw. auf Wunsch von Lesern, Kunden, Entwicklern aktuelle Themen aufsetzten und uns dort einen passenden Experten holen. Für Wünsche, Anregungen etc. sind wir wie immer offen. Es soll ein Mix aus kostenlosen und optional auch intensiveren kostenpflichtigen Schulungen (ggf. mit Material) werden.

Für die Schulungen haben wir daher eine Internetseite [2] eingerichtet, auf der man die Termine und Anmeldeformulare findet.

In diesem Sinne freuen wir uns auf viele weitere nette Gespräche und Themen.

[1] http://www.eurocircuits.com

[2] http://schulungen.embedded-projects.net □

Nächste Termine:

6.2.2013 Arbeiten mit dem Platinen-Layout Programm Eagle (Richard Hammerl)

13.03.2013 embedded GNU/Linux als Mikrocontroller-Ersatz (Benedikt Sauter)

PWM mit FPGA

Christian Kähler <kaehler.christian@gmx.net>

Einleitung

Im Rahmen einer Lehrveranstaltung an der Hochschule Merseburg (FH) ist dieses Projekt im Sommersemester 2012 entstanden. Ziel ist es, einen handelsüblichen PWM-Lüfter aus dem PC-Sektor mit einem FPGA anzusteuern und dessen aktuelle Drehzahl auszulesen (siehe Abb 1). Das Ganze soll dahingehend weiterentwickelt werden, dass die Drehzahl automatisch in Abhängigkeit von Temperaturen geregelt wird. Dieser Artikel soll zeigen, wie ein PWM-Lüfter manuell über Taster gesteuert werden kann. Es ist natürlich möglich, das Ganze auf die Steuerung mehrerer Lüfter auszuweiten. Die Projektdateien und auch eine ausführlichere Beschreibung des Projektes sind zu finden unter:

http://www.hs-merseburg.de/~8ckaehle/master/projekt_PWM_Luefter.htm

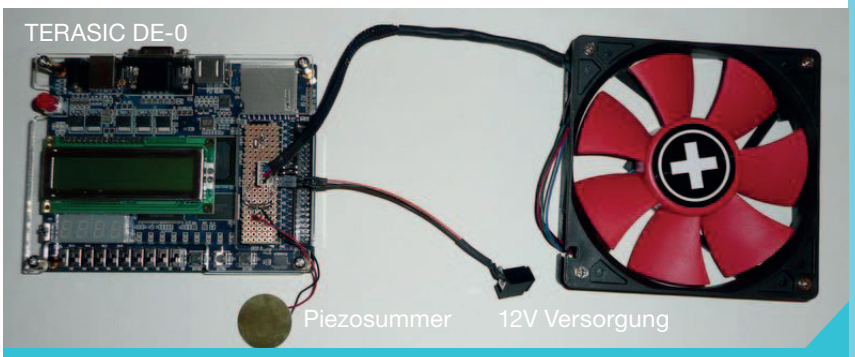


Abb. 1: Versuchsaufbau

Spezifikation der elektrischen Anforderungen

Abb. 2: Steckerbelegung PWM-Lüfter

Für den Umgang mit PWM-gesteuerten Lüftern gibt es besondere Spezifikationen, welche von Intel unter [1] veröffentlicht wurden. Diese stellen einen Standard dar. Handelsübliche Lüfter aus dem PC-Sektor müssen mit einer Versorgungsspannung von 12 V versorgt werden [vgl. 1, S.9]. Der standardisierte Steckverbinder ist vierpolig. Abbildung 2 zeigt den Stecker eines PWM Lüfters mit folgender Pinbelegung:

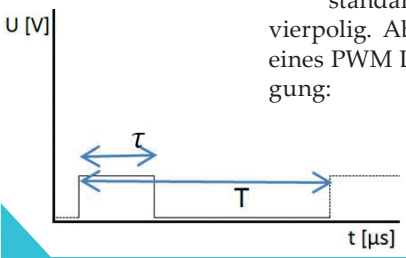


Abb. 3: PWM-Signal

```
Pin1 – PWM-Eingang
; Pin2 – Tachosig-
nal; Pin3 – +12V
; Pin4 – GND
```

Die Drehzahl des Lüfters ist proportional zu einem zu erzeugenden PWM-Signal. Hierbei hängt die Drehzahl von der Pulsbreite ab. Nominal ist eine Frequenz zwischen 22kHz und 28kHz zu erzeugen. Pro Umdrehung liefert der Lüfter zwei Pulse, die zur Erkennung der aktuellen Drehzahl ausgewertet werden können. Beim Tachosignalausgang handelt es sich um eine Open-Collector beziehungsweise eine Open-Drain Schaltung. Es ist daher zwingend ein Pullup-Widerstand vorzusehen [vgl. 1, S.9]. Abbildung 3 zeigt eine Periode des zu erzeugenden PWM-Signals. Für das Projekt ist wichtig, die Periodendauer T auf 40µs festzusetzen. Dies entspricht einer Frequenz von 25kHz. Die Länge des Pulses τ ist hierbei variabel. Aus den beiden Werten lässt sich der Duty Cycle berechnen, welcher die eigentlich wichtige Größe zur Steuerung der Drehzahl des Lüfters darstellt.

```
(DC=τT*100)
```

Generierung eines PWM-Signals aus dem Systemtakt

Für die Geschwindigkeitsregelung ist die direkte Steuerung der Pulsbreite des PWM-Signals vorgesehen. Der Duty Cylcle kann hierbei mittels Taster zwischen 30 % und 80 % in 5%-Schritten eingestellt werden. Die Erzeugung des Signals erfolgt durch eine Teilung des 50 MHz Systemtakts auf einen 25 kHz PWM-Takt mit variabler Einstellung der Pulsbreite. In der Architektur der entity „pwm“ (pwm.vhd) existieren zwei unabhängige, parallel laufende Prozesse. Der erste Prozess nimmt die Nutzereingaben entgegen und verändert die Pulsbreite des PWM-Signals. Der zweite Prozess sorgt für die Ausgabe des PWM-Signals und für die Ansteuerung der LED-Leiste.

Die Ableitung des benötigten Takts soll durch einfaches Zählen der Flanken des Systemtakts realisiert werden. Im Voraus ist der erforderliche Maximalwert des Zählers nach folgender Vorschrift zu berechnen:

```
1 s50000000 [Schritte]=40 µsx
[Schritte]    -> x=40*10-6*50
*106 =2000
```

Hieraus ergibt sich, dass ein Zähler bei einem 50MHz Takt genau 40µs benötigt um den errechneten Maximalwert von 2000 zu erreichen. Dies spiegelt die gewünschte Periodendauer direkt wider. Um ein PWM-Signal zu erzeugen, muss auf dem Ausgang, in diesem Fall der Port „pwm_out“, für eine definierte Zeit ein High-Signal ausgegeben werden.

Wie bereits erläutert, wird zur direkten Steuerung der Geschwindigkeit nur der Duty Cycle verändert. Der erste Prozess der entity „pwm“ (pwm.vhd Zeile 25-38) realisiert genau das, über einen vom Benutzer bedienbaren Taster. Der Prozess „pwm_einstellung“ reagiert hierbei auf eine fallende Flanke des Tasters (pwm.vhd Zeile 28). Eine fallende Flanke signalisiert das Drücken des Tasters, da diese auf dem verwendeten DE0 Board LOW-Aktiv angebunden sind Erste praktische Versuche am verwendeten Lüfter zeig-

ten, dass keine Geschwindigkeitserhöhung ab einem Duty Cycle von 80% stattfinden. Daher wurde der maximale Duty Cycle auf 80% begrenzt. Dies entspricht einem Zählerwert von 1600. Die Einstellung des neuen Duty Cycles erfolgt durch eine einfache Abfrage (pwm.vhd, Zeilen 29-33). Ist der aktuelle Duty Cycle beziehungsweise Zählerwert kleiner als das Maximum, wird der aktuelle Zählerstand um 100 erhöht. Dies entspricht einer Erhöhung des Duty Cycles um 5%. Ist der Maximalwert erreicht, wird der Duty Cycle auf den Wert 30% zurückgesetzt. Dies entspricht einem Zählerwert von 600. Der aktuelle Wert des Duty Cycles wird in einem weiteren Prozess direkt zur Ausgabe des PWM-Signals genutzt. Der Prozess „pwm-ausgabe“ (pwm.vhd, Zeilen 44-80) teilt sich in zwei Vorgänge. Als erstes wird anhand des im Prozess „pwm_einstellung“ eingestellten Duty Cycles ein LED-Vektor angesteuert, um die aktuelle Einstellung für den Benutzer sichtbar zu machen. In einem zweiten Schritt wird das PWM-Signal erzeugt. Das Signal „pwm_HIGH“ wird hierbei direkt genutzt, um zu bestimmen, für wie viele Zählschritte das Ausgangssignal einen High-Pegel führen soll. Mit einer einfachen Case-Anweisung wird entschieden, wie viele der zehn LEDs in Abhängigkeit von der Länge der Pulsbreite leuchten sollen (pwm.vhd, Zeilen 46-60). Die eigentliche Erzeugung des PWM-Signals realisiert der zweite Teil des Prozesses (pwm.vhd, Zeilen 63 – 74).

```
25  pwm_einstellung: process(schalter)
26      begin
27
28          if falling_edge(schalter) then
29
30              if pwm_HIGH < 1600 then
31                  pwm_HIGH <= pwm_HIGH+100;
32              else
33                  pwm_HIGH <= 600;
34              end if;
35
36          end if;
37
38      end process pwm_einstellung;
```

Abb. 4: VHD Teil 1

```
43
44  pwm_ausgabe:   process (clk50MHz, reset)
45      begin
46          if (rising_edge(clk50mhz)) then
47
48          case pwm_HIGH is
49
50              when 600 => led_vector <="0000000000";
51              when 700 => led_vector <="1000000000";
52              when 800 => led_vector <="1100000000";
53              when 900 => led_vector <="1110000000";
54              when 1000 => led_vector <="1111000000";
55              when 1100 => led_vector <="1111100000";
56              when 1200 => led_vector <="1111110000";
57              when 1300 => led_vector <="1111111000";
58              when 1400 => led_vector <="1111111100";
59              when 1500 => led_vector <="1111111110";
60              when 1600 => led_vector <="1111111111";
61              when others => led_vector <= "0000000000";
62          end case;
63
64
65          if reset = '0' then
66              pwm_out <= '0';
67          else
68              if (count1 < (PWM_freq - PWM_HIGH)) then
69                  pwm_out <= '0';
70                  count1 <= count1 + 1;
71              elsif (count1 < (PWM_freq)-1) then
72                  pwm_out <= '1';
73                  count1 <= count1 + 1;
74              else
75                  count1 <= 0;
76              end if;
77          end if;
78      end process pwm_ausgabe;
79  end rtl;
80
```

Abb. 5: VHD Teil 2

Generierung eines PWM-Signals aus dem Systemtakt

Für die Detektion und Auswertung des Tachosignals sind mehrere Prozesse und Blöcke notwendig. Wie bereits erläutert, liefert der Lüfter für jede Umdrehung zwei Pulse. Diese können für die Auswertung gezählt werden. Um eine Berechnung der Umdrehungen pro Minute zu ermöglichen, ist eine definierte Messzeit sicherzustellen. In diesem Fall bietet es sich an, die Pulse eine Sekunde lang zu zählen, da hierdurch leicht eine Umrechnung auf die Umdrehungszahl pro Minute ermöglicht wird. Für die Erzeugung eines Pulses pro Sekunde ist ein einfacher Zähler ausreichend. Der Zähler inkrementiert seinen Wert im Bereich von 0 – 49999999 bei jeder steigenden Flanke des Systemtakts (sek_count.vhd, Zeilen 20-25). Erreicht der Zähler einen Stand von 49999999, gibt er ein Flag aus, welches das Erreichen einer Sekunde signalisiert (sek_count.vhd, Zeilen 26-28). Anhand dieses Flags kann die Auswertung angestoßen werden. Durch den Auswertungsprozess wird das Erkennen dieses Flags bestätigt, wodurch der Zähler auf seinen Minimalwert zurückgesetzt wird (sek_count.vhd, Zeilen 21-23). Die eigentliche Auswertung und Anzeige des Tachosignals realisieren zwei weitere Prozesse, welche in einem Block enthalten sind. Die entity „tacho“ (tacho.vhd) enthält einen Prozess, der die Pulse zählt und nach einer Zählzeit von einer Sekunde die Umrechnung in die Umdrehungszahl pro Minute anstößt. Ein zweiter Prozess in dieser Architektur führt die Berechnung und die Ausgabe der aktuellen Umdrehungszahl durch. Die Pulse werden mit einer Abtastfrequenz von 50 MHz, also dem Systemtakt, gezählt (tacho.vhd, Zeile 39). Da innerhalb eines Prozesses nur auf die Flanken eines Signales getriggert werden kann, muss die Erfassung der Pulsflanken des Lüfters durch eine andere Methode realisiert werden. Ein einfaches zwei Bit breites Schieberegister eignet sich hier in besonderem Maße (tacho.vhd, Zeilen 40-41). Wird eine Flanke erkannt (flanke = „10“) und die Messzeit einer Sekunde ist noch nicht erreicht (flag = 0), wird der Pulszähler „count“ inkrementiert (tacho.vhd, Zeile 43-45). Ist die maximale Messzeit erreicht (flag = 1), wird der Zähler zurückgesetzt und die Auswertung angestoßen. Zu jeder steigenden Flanke des Systemtaktes wird zunächst das Clear-Signal zurückgesetzt (tacho.vhd, Zeile 63/64). Wird dieses Signal gesetzt, wird der Sekundenzähler zurückgesetzt (sek_count.vhd, Zeile 21) und somit die Fertigstellung der Verarbeitung signalisiert. Führt das Signal „flag“ des Sekundenzählers einen High-Pegel, muss die Verarbeitung der gemessenen Pulsflanken gestartet werden (tacho.vhd, Zeile 65). Für die Umrechnung der in einer Sekunde gezählten Pulse in die Umdrehungszahl pro Minute wird folgende Beziehung verwendet:

```
35  tacho_zaehl:   process (clk50MHz ,tacho_in)
36      --variable mittelung : integer range 0 to 10 :=0;
37
38      begin
39          if (rising_edge(clk50MHz)) then
40              flanke(1)<=flanke(0);
41              flanke(0)<=tacho_in;
42
43              if (flanke = "10" and flag = '0') then
44                  count <= count + 1;
45                  cnt_temp <= count+1;
46              end if;
47
48              if flag = '1' then
49                  count <= 0;
50              end if;
51          end if;
52  end process tacho_zaehl;
```

Abb. 6: Tacho VHD Teil 1

```
55  tacho_auswert : process (clk50MHz,flag)
56      variable wert : integer range 0 to 9000;
57      variable temp : integer;
58      variable tausender : integer range 0 to 9;
59      variable hunderter : integer range 0 to 9;
60      variable zehner    : integer range 0 to 9;
61      variable einer     : integer range 0 to 9;
62  begin
63      if (rising_edge(clk50MHz) ) then
64          clear <= '0';
65          if flag = '1' then
66              -- ticks in rpm umrechnen
67              wert := (cnt_temp * 30);
68              if ((cnt_temp*30) < 200) then
69                  alarm_out <= '1';
70              else
71                  alarm_out <= '0';
72
73              end if;
74              -- Einerstelle extrahieren
75              temp := wert mod 10;
76              einer := temp;
77
78              -- Zehnerstelle extrahieren
79              wert := wert - temp;
80              temp := wert mod 100;
81              zehner := temp / 10;
82
83              -- Hunderterstelle extrahieren
84              wert := wert - temp;
85              temp := wert mod 1000;
86              hunderter := temp / 100;
87
88              -- Tausenderstelle extrahieren
89              wert := wert - temp;
90              temp := wert mod 10000;
91              tausender := temp / 1000;
92              clear <= '1';
93          end if;
94      end if;
95      ein <= conv_std_logic_vector(einer, 4);
96      zehn <= conv_std_logic_vector(zehner, 4);
97      hundert <= conv_std_logic_vector(hunderter, 4);
98      tausend <= conv_std_logic_vector(tausender, 4);
99  end process tacho_auswert;
100 end rtl;
```

Abb. 6: Tacho VHD Teil 2

```
RPM=Pulses*602=Pulses*30
```

In der VHDL-Beschreibung findet sich der errechnete Wert schlussendlich in der Variable „wert“ (tacho.vhd, Zeile 67). In einem ersten Schritt wird überprüft, ob die aktuelle Drehzahl sich in einem kritischen Bereich befindet. Kritisch wird hierbei als „Umdrehungen < 200“ definiert. Ist dies der Fall, wird ein Flag gesetzt, welches in einem weiteren Prozess einen Piezosummer als Alarmgeber ansteuert (tacho.vhd, Zeilen 68-71). Durch die Berechnung (tacho.vhd, Zeile 67) entsteht ein vierstelliger Dezimalwert. Da dieser nicht ohne weitere Umformung auf den Sieben-Segment-Anzeigen ausgegeben werden kann, bedarf es einer weiteren Umformung. Mittels Modulo-Operatoren wird die jeweilige Stelle aus der Dezimalzahl extrahiert (tacho.vhd, Zeilen 74-92). Abgeschlossen wird die Verarbeitung mit der Signalisierung für den Sekundentimer (clear = 1).

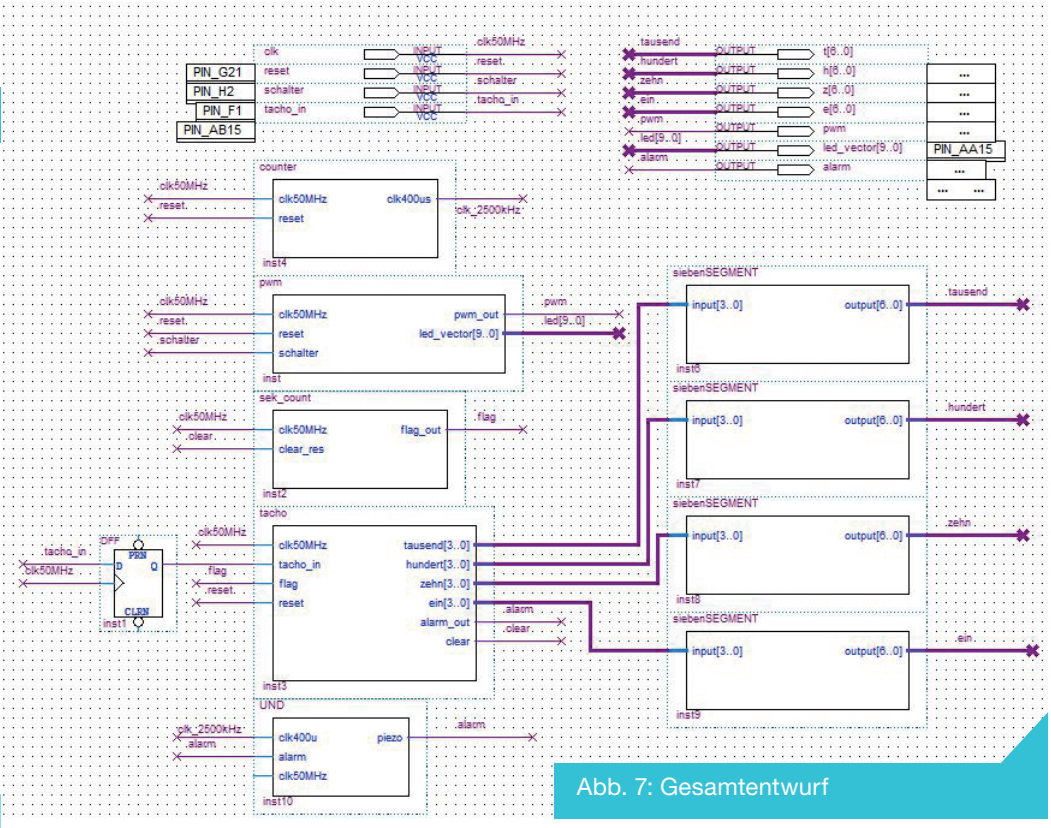


Abb. 7: Gesamtentwurf

Literatur

[1] Intel Corporation (2005): 4-Wire Pulse Width Modulation (PWM) Controlled Fans – Specification – Revision 1.3 Online im Internet. URL: http://formfactors.org/developer/specs/4_Wire_PWM_Spec.pdf (Abruf: 12.04.2012)

Ethernet-Schnittstelle für das STM32F4 Discovery Board

Simon Tewes, Christoph Budelmann <mail@budelmann-elektronik.com>

Einleitung

Die Firma ST Microelectronics bietet mit dem STM32F4 Discovery Board [1] ein sehr günstiges Entwicklungsboard mit interessanter Peripherie und einem sehr leistungsfähigen Mikrocontroller an. Das Board ist ideal für erste Schritte mit einem ARM Cortex-M4 basierten Mikrocontroller. In diesem Artikel soll das Board kurz vorgestellt sowie die Erweiterung des Boards um eine Ethernet-Schnittstelle beschrieben werden, die auf dem Basisboard nicht vorhanden ist. Zum Test der Hardware dient dabei ein angepasstes Demo-Projekt von ST Microelectronics, über das die Leuchtdioden des Basisboards über eine Web-Oberfläche gesteuert werden können.

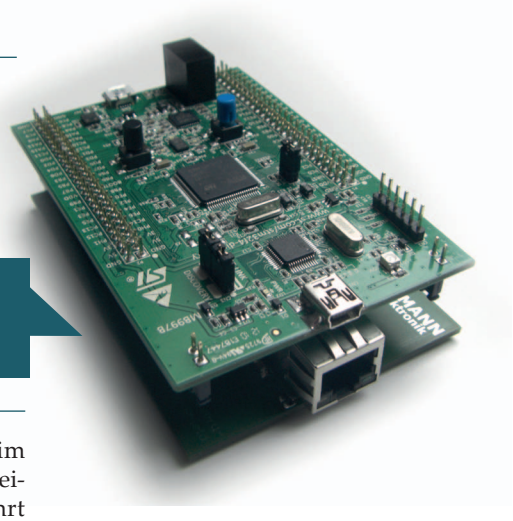


Abb. 1: STM32F4 Discovery Board

Hardware

Kern des STM32F4 Discovery Boards ist der STM32F407VG Mikrocontroller im LQFP100-Gehäuse, der mit einer maximalen Taktfrequenz von 168MHz arbeitet und dessen Pins vollständig auf zwei zweireihige Stiftleisten herausgeführt sind. Damit lässt sich das Board einfach und schnell um eigene Peripherie erweitern. Ein Programmieradapter ist direkt auf dem Board integriert, so dass keine weiteren Adapter nötig sind um mit der Programmierung zu starten. Der sogenannte ST-LINK/V2 nutzt die SWD-Schnittstelle (Serial Wire Debug) des Mikrocontrollers. Die Verbindung mit dem PC erfolgt über eine Mini-USB-B-Buchse, die das Board auch mit Strom versorgen kann. Daneben sind auf dem Board ein dreiachsiger MEMS-Beschleunigungssensor, ein MEMS-Mikrofon, ein Audio-DAC mit Class-D Verstärker, acht Leuchtdioden, zwei Taster und ein Mikro-USB OTG Anschluss vorhanden.

Entwicklungsumgebung

Eine sehr interessante Hardware-Plattform nützt natürlich nichts, wenn keine passende Entwicklungsumgebung verfügbar ist. Im Folgenden möchten wir hier exemplarisch auf die Open-Source Entwicklungsumgebung CooCox [2] eingehen, die sich insbesondere für Einsteiger in die Welt der ARM-Programmierung sehr gut eignet.

CooCox CoIDE ist eine Entwicklungsumgebung, die auf Eclipse [3] basiert und viele Voreinstellungen bei der Installation bereits automatisch vornimmt. Ab der Version V1.4.1 unterstützt CoIDE die neuen ARM Cortex-M4 Mikrocontroller und bereits seit längerem den ST-LINK/V2. Außer der Toolchain für die ARM-Mikrocontroller wird keine weitere Software benötigt. Als Toolchain soll die GNU Toolchain for ARM Embedded Processors [4] verwendet werden, deren aktuelle Version die 4.6_2012q2 ist.

Als Erstes sollte die GNU-Toolchain installiert werden und dann die CooCox-Tools. CooCox bietet hierfür das CoCenter an, über das die CoIDE und CoFlash installiert werden können. CoFlash dient zum Übertragen der erzeugten Binär-Datei auf den Mikrocontroller.

Ist die Installation abgeschlossen müssen noch einige wenige Einstellungen gemacht werden und es kann losgehen.

Zunächst startet man CoIDE und wählt in der Menüleiste

```
/Project/ Select Toolchain Path
```

Dort wird der Pfad zum „bin“ Ordner der GNU-Toolchain eingetragen. Anschließend muss nur noch der Debugger eingestellt werden, nachdem ein Projekt angelegt wurde.

Dafür wählt man

```
/Project/New Project
```

in der Menüleiste aus, gibt dem Projekt einen Namen und wählt den Speicherort.

Auf der Startseite befindet sich nun der Reiter „Repository“, sollte dieser sich dort nicht befinden, kann man diesen unter „View“ aufrufen. Im nun folgenden Dialog werden die Rahmenbedingungen des neuen Projektes festgelegt. Für das STM32F4 Discovery Board wählt man

```
>>ST>>STM32F407VG
```

als Mikrocontroller aus. Auf den nun folgenden Seiten können Bibliotheken gewählt werden, die für das Projekt verwendet werden. Bei diesen Komponenten handelt es sich um die Standard-Bibliotheken von ST Microelectronics.

Als letzter Schritt folgt die Einstellung des Debuggers, um das Übertragen der Binärdatei auf das Board zu ermöglichen. Man öffnet in CoIDE den Reiter „Debug“ in der Menüleiste und wählt „Debug Configuration“. Unter „Cortex-M Application“ befindet sich ein Punkt „XXX. configuration“ der den vergebenen Projektnamen trägt. Dort trägt man unter Adapter den ST-Link/V2 ein und wählt als Port SWD. Nun kann das erste Programm geschrieben oder ein Beispielprojekt geöffnet werden.

ST Microelectronics liefert sehr umfangreiche Bibliotheken und Beispiele, diese können auf der ST Microelectronics Homepage unter „Design Support“ für das STM32F4 Discovery Board heruntergeladen werden. In den Bibliotheken beziehungsweise Beispielen finden sich Beispiele für fast alle Peripheriekompo-

nenten des Discovery Boards und erleichtern somit einen Einstieg ungemein, zumal sie einem viel Recherche im doch sehr umfangreichen Datenblatt zum STM32F4 ersparen können.

Ethernet

Um das Discovery Board Ethernet fähig zu machen bedarf es einiger zusätzlicher Hardware, welche in Form eines Aufsteckboards beziehungsweise „Shields“ realisiert wurde. Da der STM32F407VG bereits einen Media Access Controller (MAC) integriert hat, beschränkt sich die zusätzliche Hardware auf die physikalische Schnittstelle, dem Phy und einer RJ45-Buchse mit Übertrager. Als Ethernet-Phy wurde der DP8348 gewählt, zudem befindet sich auf dem Shield noch eine RJ45-Buchse mit integriertem Übertrager und zwei Leuchtdioden. Die Spannungsversorgung erfolgt über das Discovery Board. Grundsätzlich gibt es bei der Ansteuerung des Phy zwei Möglichkeiten, einmal per MII (Media Independent Interface) und per RMII (Reduced Media Independent Interface). MII nutzt insgesamt acht Leitungen mehr als RMII, allerdings wird es auch nur mit der halben Taktrate von RMII betrieben. Da einige für MII benötigte Pins jedoch von den Peripheriebauteilen des Discovery Boards bereits belegt sind, wurde der Ethernet-Phy über RMII angebunden.

Schaltplan

Abbildung 2 zeigt den Schaltplan des Shields. Die Außenbeschaltung des Ethernet-Phy besteht dabei lediglich aus einer Hand voll passiver Bauteile und dem Quarzoszillator zur Erzeugung eines 50MHz Taktes.

Software

ST Microelectronics bietet auf seiner Homepage ein Ethernet-Beispiel namens „Webpage Demonstration“ an, das für das STM3240G-Eval-Board geschrieben wurde und sich sehr einfach an das Discovery Board anpassen lässt. Ein Projektordner inklusiver aller Anpassungen für das Discovery Board und die CoIDE Anpassung können dem Blog von TKJ Electronics [5] entnommen werden.

Das Projekt umfasst einen kleinen Webserver, der eine Website ins Netzwerk stellt und über diese einige Funktionen des Boards zugänglich macht. Es ist dabei sowohl möglich, die IP-Adresse des Boards fest einzustellen als auch DHCP zu nutzen. Die Einstellung hierzu findet sich in der main.h. Über die Eingabe der IP-Adresse im Browser wird die Website aufgerufen und es können über diese die Leuchtdioden ein- und ausgeschaltet sowie ein AD-Wandler-Kanal ausgelesen werden.

Zusammenfassung

Sowohl für Neueinsteiger in die Welt von Mikrocontrollern als auch für Umsteiger, die die Leistungsfähigkeit von 32bit Mikrocontrollern erforschen wollen, eignet sich das Discovery Board in idealer Weise. Erfahrungen im Einrichten einer ARM-Toolchain und IDE sind bei der Verwendung der CooCox Umgebung ebenfalls nicht nötig, womit schnelle erste Erfolge garantiert sein sollten. Mit der nachgerüsteten Ethernet-Schnittstelle eignet sich das Discovery Board aber auch für anspruchsvollere Mess-, Steuer- und Regel-Anwendungen im Netzwerk oder auch als einfacher Webserver.

Literatur

[1] STM32F4 Discovery Board: http://www.st.com/internet/evalboard/product/252419.jsp

[2] CooCox Entwicklungsumgebung: http://www.coocox.org

[3] Eclipse: http://www.eclipse.org

[4] GCC Toolchain for ARM Embedded: https://launchpad.net/gcc-arm-embedded

[5] Angepasstes „Website Demonstration“ Projekt für CooCox: http://blog.tkjelectronics.dk/2012/08/ethernet-on-stm32f4discovery-using-external-phy/

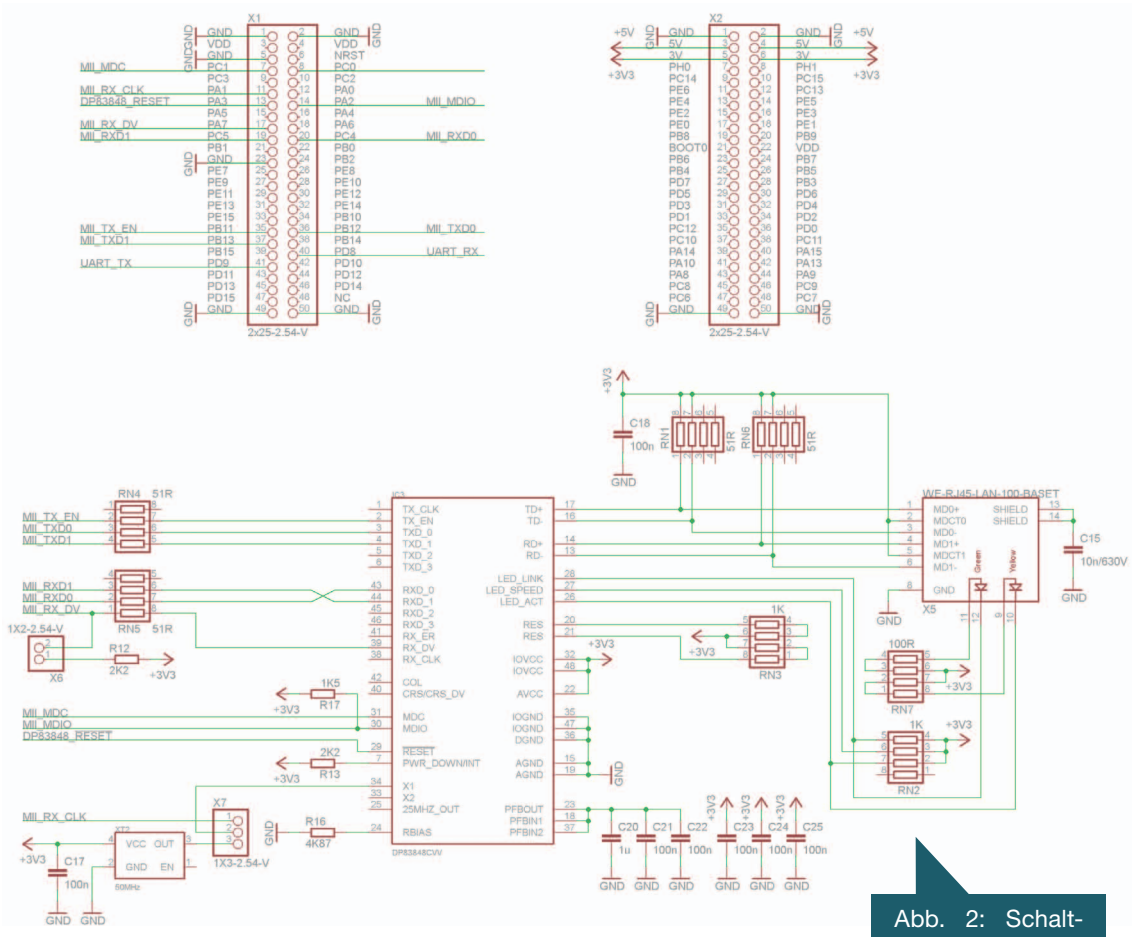


Abb. 2: Schaltplan des Shields

USB-CAN Adapter

Praktikumsbericht

Pamela Bogner <bogner@hm.edu>, Christian Truckenbrodt <ct@lessmueller.de>

Seit September bin ich Praktikantin bei Lessmüller Lasertechnik in München und absolviere dort mein Praxissemester. Lessmüller Lasertechnik stellt Prozessüberwachungssysteme für das Laserstrahlschweißen her. Um in zukünftigen Projekten die Evaluierung von CAN-Bus-Sensoren via USB vornehmen zu können, wurde mir die Aufgabe zugeteilt eine Hard- und Software zu entwickeln, die dies einfach und zugleich flexibel ermöglicht.

Die Hardware

Um nicht viel Zeit mit dem Layouten von Platinen zu verbringen, wird die Hardware aus zwei fertigen Entwicklungsboards zusammengesteckt.

Das zentrale Herzstück des USB-CAN Adapters bildet der CAN-Controller MCP2515 von Microchip. Hierbei handelt es sich um einen CAN-Controller, der über eine SPI Schnittstelle angesprochen wird und CAN Nachrichten senden und empfangen kann. Er wird mit 5V versorgt. Zudem wird ein CAN-Transceiver wie der MCP2551 zwischen der Busleitung und dem CAN-Controller verwendet.

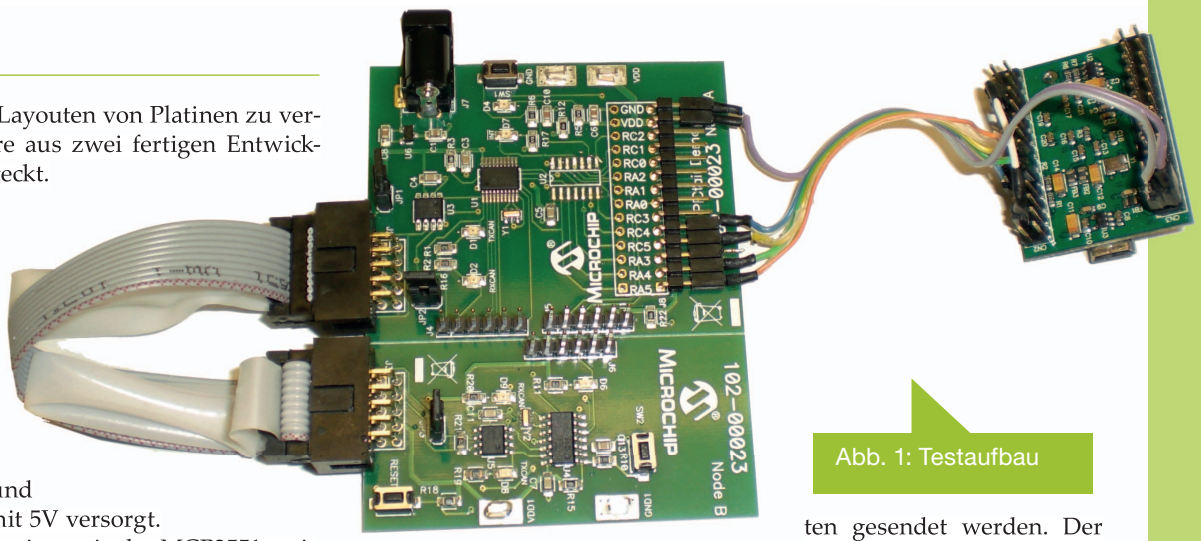
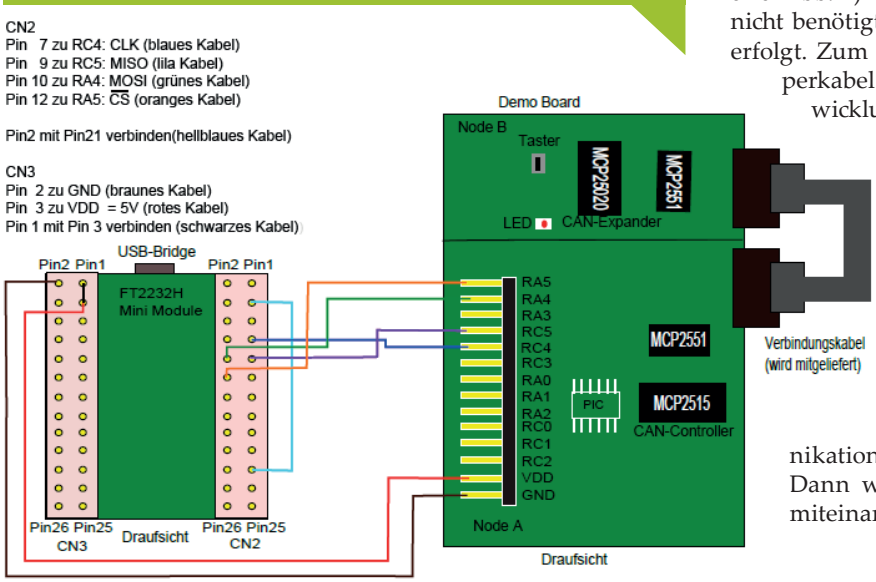


Abb. 1: Testaufbau

Beide ICs finden sich auf dem MCP2515 PictailTM Demo Board von Microchip [1]. Außerdem enthält das Board eine abtrennbare Platine mit dem CAN-Expander MCP25020, einer LED und einem Taster. Der MCP25020 ist bereits vorprogrammiert und kann zum Testen der CAN Kommunikation eingesetzt werden. Zum Einschalten der LED muss eine CAN Message an den CAN-Expander mit ID, Länge, Adresse, Maske und Daten gesendet werden. Der Zustand des Tasters kann durch eine Request Message*, die aus ID und Datenlänge besteht, ermittelt werden.

Da die meisten PCs nicht über eine SPI Schnittstelle verfügen wird das FT2232H Mini Module von FTDI [2] als USB Bridge eingesetzt. Dies ist ebenfalls als Entwicklungsboard mit allen benötigten Bauteilen wie z.B. Spannungsregler, Oszillator, etc. erhältlich. Die USB Bridge kann je nach verwendeter SW-Bibliothek als UART, JTAG, I²C und SPI Bridge verwendet werden. Zudem verfügt sie über mehrere Kanäle.

Abb. 2: Verdrahtung der beiden Boards



Die Verdrahtung der beiden Entwicklungsboards (siehe Abb. 1 und Abb. 2) ist relativ einfach. Ein zusätzliches Netzteil wird nicht benötigt, da die Versorgung über den USB-Port des PCs erfolgt. Zum Verbinden der einzelnen Steckpins werden Jumperkabel eingesetzt. Zunächst muss auf dem FTDI Entwicklungsboard Pin1 (CN3) mit Pin 3 (CN3) verbunden werden, damit der 3,3V Spannungsregler mit der 5V Stromversorgung des USB Busses versorgt wird. Weiterhin ist es notwendig Pin3 (CN2) mit Pin21 (CN2) zu verbinden. Diese Verbindung sorgt für die richtige Betriebsspannung von 3,3V für VCCIO auf dem FT2232H Chip.

Anschließend wird auf dem MCP2515 PictailTM Demo Board der Microcontroller PIC16F676 herausgelötet, damit er die Kommunikation zwischen MCP2515 und FT2232H nicht stört. Dann werden die SPI-Pins MISO, MOSI, CLK und CS miteinander verbunden. Eine Kreuzung der Datenpins

wie beispielsweise bei UART, ist bei SPI nicht nötig. Zur Versorgung des CAN-Controllers und des CAN-Transceivers muss eine weitere Leitung mit 5V (Pin3 CN3 zu VDD) verbunden werden. Natürlich braucht es abschließend noch eine GND Verbindung.

Die Software

Die Software besteht aus einem einfachen Konsolenprogramm, das sich aus drei Ebenen zusammensetzt.

- SPI Ebene: Von FTDI gibt es eine fertige Bibliothek [3] anhand derer die SPI Funktionen des Mini Modules mit wenigen Befehlen (read, write, open, close, get channels, init channels …)genutzt werden können
- MCP2515 Ebene: Des Weiteren enthält das Konsolenprogramm die MCP2515 Bibliothek von Fabian Greif [4]. Hier sind allerdings einige Anpassungen notwendig, da die Bibliothek ursprünglich für Mikrocontroller und nicht für PC basierte Anwendungen geschrieben wurde. In der Bibliothek findet man Funktionen mit denen man Register auslesen und beschreiben kann, aber auch den Mikrocontroller initialisieren und in verschiedene Zustände (listen, sleep ...) setzen kann.
- CAN Ebene: Zur komfortablen Nutzung des CAN Adapters in eigenen Programmen gibt es die folgenden Funktionen: CAN: init, set ID, send data, get message, set mode, close. Sie greifen allesamt auf Funktionen der MCP2515 und FT2232 Bibliotheken zu.
- Im Beispielprogramm (siehe rechts) wird im Anschluss an eine grundsätzliche Initialisierung das Konsolenprogramm fortlaufend in einer While-Schleife ausgeführt. Es wird geprüft, ob die Taste auf dem Expander Board gedrückt wurde und sendet ggf. eine CAN Message zum Ein- oder Ausschalten der LED.

  Der TXB0 Sendepuffer wird dauerhaft mit der ID 0x500 beschrieben. Somit ist es ausreichend zum Schalten der LED lediglich die data[ ] Bytes zu manipulieren und das gesamte Paket aus dem Sendepuffer TXB0 zu verschicken. Erhält der Expander eine CAN Message mit der ID 0x500 und data[2] = 0x00, so wird die LED eingeschaltet. Ist data[2] = 0x10, dann schaltet die LED wieder ab.

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

#include „can.h"

void main(void)
{
   can _ t msg;
   uint8 data[8];
   data[0] = 0x1E;     // Adresse des zu beschreibenden
   Register
   data[1] = 0x10;     // Maske

   if (can _ init(BITRATE _ 125 _ KBPS))
   {
     // Empfangspuffer leeren
     can _ get _ message(&msg);

     // CAN ID 0x500 in Sendepuffer TXB0 schreiben
     can _ set _ id(0x500, 0, TXB _ 0);     // ID 0x500, keine
     // extended ID, Puffer TXB0

     // CAN ID 0x280 in Sendepuffer TXB1 schreiben
     can _ set _ id(0x280, 0, TXB _ 1);     // ID 0x280, keine
     // extended ID, Puffer TXB1

     do
     {
            // Nachricht aus Sendepuffer TXB1 versenden
            can _ send _ data(0, 8, Rtr, TXB _ 1); // keine
            // Daten, Länge 8, RTR Nachricht, Puffer TXB1

            // Empfangene Nachrichten abfragen
            can _ get _ message(&msg);

            // empfangene Daten auswerten
            if((msg.data[1] & 0x01) == 0x00)
            // wenn Taste gedrueckt, LED an
            {
                   data[2] = 0x00;
                   printf(„Taste gedrueckt, LED an\n");
            }
            else
            // sonst LED aus
            {
                   data[2] = 0x10;
                   printf(„Taste nicht gedrueckt, LED
                   aus\n");
            }
            // Nachricht senden, ob LED an oder aus
            can _ send _ data(data, 3, 0, TXB _ 0); // Daten,
            //Länge 3, keine RTR Nachricht, Puffer TXB0

     } while(! _ kbhit()); // zyklisch fortlaufend bis
     //belibige Tastatureingabe kommt

     can _ close();
   }
}
```

Wenn der Expander eine CAN Message mit der ID 0x280 und einem gesetzten RTR Flag empfängt, wird er mit der geforderten Datenanzahl antworten. Im Bespielprogramm mit 8 Byte, das entspricht Daten aus 8 Registern. Um den Tastendruck abzufragen, wird in diesem Fall das 2. Register ausgelesen, wenn in diesem 0x01 drinsteht wurde die Taste nicht gedrückt bei 0x00 wurde sie gedrückt.

Fazit

Das beschriebene Projekt stellt eine schnelle und sehr flexible Möglichkeit dar einen USB-CAN Adapter selber zu bauen. Der Aufbau der Hardware ist auf Grund der verwendeten Entwicklungsboards sehr gering. Die Software ermöglicht einen unbegrenzten Zugriff auf alle Funktionen des CAN-Controllers. Somit ist es mit etwas programmieren möglich weitere Projekte wie z.B. einen USB-CAN-Logger aufzubauen. Das Praktikum bei Lessmüller Lasertechnik war eine interessante Herausforderung für mich. Ich konnte mich in den Bereichen Elektronik und Programmieren weiterbilden. Dies geschah nicht nur durch das Projekt, sondern auch durch interessante Gespräche mit den anderen Studenten und Mitarbeitern. Zudem hat es Spaß gemacht an einem abgeschlossenen Projekt zu arbeiten, welches sich später in den Produkten wieder finden wird.

Links/ Download

[1] http://ww1.microchip.com/downloads/en/DeviceDoc/51572a.pdf

[2] http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_FT2232H_Mini_Module.pdf

[3] http://www.ftdichip.com/Products/ICs/FT2232H.htm (Datenblatt, Driver, DLL)

[4] http://www.kreatives-chaos.com/artikel/universelle-can-bibliothek

[5] USB-CAN-Adapter.zip im Forum (siehe Einleitung S. 2 im Journal)

□

Vom Überweisungsauftrag zur TAN

Andreas Schiermeier

In den letzten Jahren sind viele Banken und Sparkassen dazu übergegangen, iTAN-Listen auf Papier durch transaktionsgebundene mTAN (TAN per SMS auf das Handy) oder TAN-Generatoren abzulösen. Diese werden je nach Institut unter den Bezeichnungen Sm@artTAN, chipTAN, oder auch „Flicker TAN“ beworben. Stets wird dabei pro Auftrag eine „flackernde“ Grafik zum Einlesen mit dem bereitgestellten TAN-Generator dargestellt. Dieser zeigt nach dem erfolgreichen Einlesen die Transaktionsdaten an und liefert nach deren Bestätigung durch den Benutzer eine TAN. Die Spezifikation der dahinter stehenden Vorgänge ist nicht allgemein einsehbar. Dieser Artikel beschreibt gemachte Beobachtungen dieses Verfahrens hinsichtlich der optischen Schnittstelle zwischen Flickergrafik und Lesegerät sowie die Smartcardkommunikation. Denkbar wäre, diese Technik (insbesondere die damit, zumindest deutschlandweit, gut verbreiteten Generatoren anderweitig zu nutzen. Über Ergänzungen, Anmerkungen oder Korrekturen freuen wir uns.

Ausgangspunkt

Es wird eine Überweisung von 1,23 EUR an die Kontonummer 335554 im Onlinebanking beauftragt. Es muss der angezeigte „Flickercode“ mit dem TAN-Generator bei eingelegter Bankkarte und nach Betätigung der Taste F eingelesen und die im Display angezeigten Transaktionsparameter, Kontonummer und Betrag, bestätigt werden. Schlussendlich zeigt der TAN-Generator die TAN 251067 zur Eingabe im Onlinebanking. Es ist zu beobachten, dass bei wiederholter Einlese- & Bestätiungs- prozedur unterschiedliche TANs erzeugt werden. Werden auf diese Weise zuviele TANs erzeugt, jedoch nicht im Online-Banking verwendet, fordert das Bankssystem eine Neusyncronisation mit der Karte an.

Frage: Wie kommt der TAN-Generator zur Anzeige der TAN 251067?

Onlinebanking Interface

Flickercode Anzeige: Das Onlinebanking zeigt einen „Flickercode“. Dies kann über unterschiedliche Techniken erfolgen:

- Adobe Flash Film
- Javascript & DOM
- animiertes GIF
- Java-Applet
- innerhalb einer nativen Applikation (z.B. StarMoney)

Die Auswahl der Darstellungstechnik richtet sich den verfügbaren Browsererweiterungen und Clienttechnologien (z.B. Terminalserver). Wenn möglich wird dem Flash Film der Vorzug gegeben.

Verbrauchertipp :)

Gelingt das Einlesen der Grafik mit dem TAN-Generator nicht oder nur sehr unzuverlässig kann es helfen die Anzeigehelligkeit am Monitor/Display und Flickergeschwindigkeit der Grafik zu variieren. Ist die Anzeigehelligkeit zu hoch erkennt der Generator selbst schwarze Bereiche als „hell“; ist sie zu niedrig können entsprechend helle Balken nicht mehr differenziert werden. Einzelne Anzeigetechniken des Flickercodes (z.B. das üblicherweise verwendete Flash) erlauben desweiteren die Flickergeschwindkeit anzupassen. Ein Klick auf den ganz linken der fünf „flackernden“ Balken verlangsamt den Ablauf der Sequenz. Dem gegenüber beschleunigen Klicks auf den Balken ganz rechts. So kann ein Kompromiss zwischen schnellem Einlesen der Daten und der Toleranz gegenüber Anzeigeproblemen (Browser, Grafiksystem, Monitor. etc) gefunden werden.

Flickercode Aufbau

Der Flicker-Code wird aus folgenden Eingabedaten generiert (im Folgenden „Flickerdaten“ genannt):

```
0F 04 87 11 00 03 03 33 55 54 14 31
2C 32 33 1D
```

Diese, bzw. eine entsprechende, Zahlenreihe findet sich auch im Quelltext der Onlinebanking-Seite als Parameter für den Flash Film oder die Javascript- & DOM-Routine.

Die Flickerdaten werden beispielsweise wie folgt zusammengestellt:

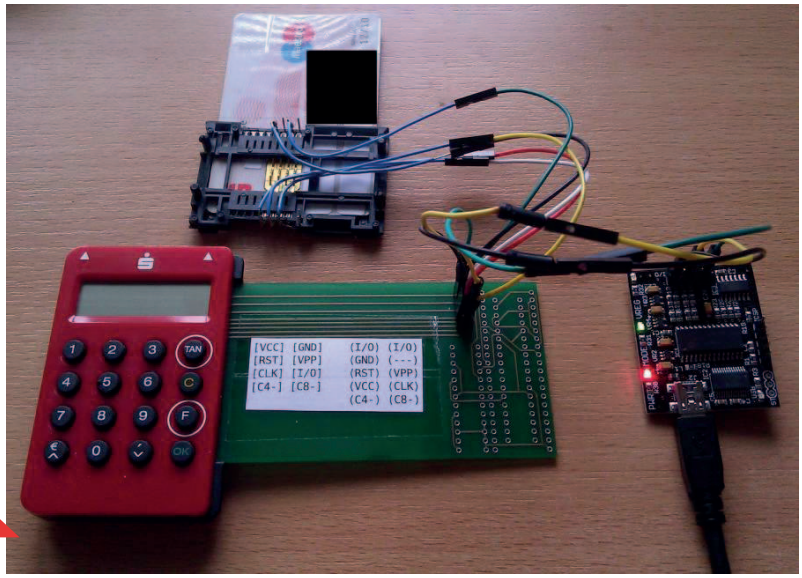


Abb. 1: Testaufbau

```
0F: Anzahl nachfolgender Bytes (15 Bytes)

0 : Kennzeichen der Kodierung für die nachfolgenden Daten in BCD (0=BCD, 1=ASCII)
 4: Länge des sog. Start-Codes in Bytes (der Start-Code definiert die vom TAN-Generator ange-
zeigten Masken)

87 11 00 03: 4 byte Start-Code in BCD-Notation - S
8          : signalisiert freie Maskengestaltung anhand nachfolgender Codes (hier 7 und 1)
 7         : signalisiert Maske „Kontonummer"
   1       : signalisiert Maske „Betrag"
    1 00 03: Übergabe einer vom Bankensystem vorgegebenen Zufallszahl (ändert mit jeder On-
line-Transaktion)

0 : Kodierung der nachfolgenden Daten in BCD
 3: Länge des ersten Maskeninhalts in Byte

33 55 54: 3 Byte: erster Maskeninhalt (Kontonummer) in BCD-Notation - M1

1 : Kodierung der nachfolgenden Daten in ASCII (wg. Komma in 1,23)
 4: Länge des zweiten Maskeninhalt in Bytes

302C3031: zweiter Maskeninhalt (hier: Betrag) in ASCII-Kodierung 1,23 - M2
31       : ASCII (hex): 1
  2C     : ASCCI (hex): , (Komma)
   32    : ASCII (hex): 2
    33: ASCII (hex): 3

1 : Luhn Prüfsumme über Start-Code und Maskeninhalte
 D: XOR Prüfsumme über die gesamten Flickerdaten ohne diese beiden Prüfsum-
menhalbbytes
```

Hinweis:
Eine ungerade Anzahl an BCD-Stellen (1/2 Byte) in Maskenwerten wird mit einem „F" als Füllzeichen zu einem vollen Byte aufgefüllt.

Luhn-Prüfsumme

Bildung der Luhn-Prüfsumme über Start-Code S und Maskeninhalte M1 & M2:; Vorgehen:

1) Jedes linke Halbbyte zählt einfach, jedes rechte Halbbyte zählt doppelt
2) Aus allen einzelnen Werten von 1. werden jeweils Quersummen gebildet und diese summiert
3) Die Prüfsumme ist 10-(Summe aus 2. modulo 10)

```
 S----------|M1------|M2---------
Am Beispiel  :  87 11 00 03 33 55 54 31 2C 32 33:
1xl          :  8 +1 +0 +0 +3 +5 +5 +3 +2 +3 +3
Quersumme 1xl:  8 +1 +0 +0 +3 +5 +5 +3 +2 +3 +3 =33
2xr          :  14+ 2+ 0+ 6+ 6+10+ 8+ 2+24+ 4+ 6
Quersumme 2xr:   5+ 2+ 0+ 6+ 6+ 1+ 8+ 2+ 6+ 4+ 6=46
-------------------------------------------------------
Summe:                                              79
10-(Summe modulo 10):                                1
=======================================================
```

XOR

Bildung der XOR-Prüfsumme am Beispiel:

```
0F 04 87 11 00 03 03 33 55 54 14 31 2C 32 33
```

Die Operation erfolgt in Übertragungsreihenfolge (erst niederwertiges Halbbyte, dann höherwertiges, 0F->F0, 0440, …)

```
F0 40 78 11 00 30 30 33 55 45 41 13 C2 23 33
```

Berechnung

```
0x0 ^ 0x4 ^ 0x0 ^ 0x7 ^ 0x8 ^ 0x1 ^ 0x1 ^ 0x0
^ 0x0 ^ 0x3 ^ 0x0 ^ 0x3 ^ 0x0 ^ 0x3 ^ 0x3 ^
0x5 ^ 0x5 ^ 0x4 ^ 0x5 ^ 0x4 ^ 0x1 ^ 0x1 ^ 0x3
^ 0xC ^ 0x2 ^ 0x2 ^ 0x3 ^ 0x3 ^ 0x3
= 0xD
```

Flickercode Übertragung

Die optischen Eingänge des TAN-Generators nehmen ein Taktsignal und je ein Halbbyte (4 bit, auch genannt Nibble) an Daten entgegen. Die Abfolge der Signalfelder ist wie folgt (auf dem Bildschirm).

Feld 1	Feld 2	Feld 3	Feld 4	Feld 5
Takt	2^0	2^1	2^2	2^3

Der eigentlichen Übertragung wird eine statische Synchronisationssequenz vorgelagert:

Feld 1	Feld 2	Feld 3	Feld 4	Feld 5
1	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	1	1	1	1
1	1	1	1	1
0	1	1	1	1
1	1	1	1	1

Flicker zum „Mitschreiben“:

1) Flicker-GIF von Onlinebanking-Service erzeugen lassen

- wget –no-check-certificate -Oflicker.gif “https://Homebankingserver/InternetPortal-Struts/CreateFlickerImage?data=...&fps=10“

  -> z.B. wget –no-check-certificate -Oflicker.gif “https://bankingportal.frankfurter-sparkasse.de/InternetPortalStruts/Creat

2) herunterladen eFlickerImage?data=0F0487110003033355541431 2C32331D &fps=10“

3) mit „gimp“ öffnen und Layer-Ansicht anzeigen Abbildung 2

Funfact/Eselsbrücke: Betrachtet man nur die Felder 2-5 bei wechselndem Takt: 0x0 0xf 0xf -> hex-speak: „Off“, gefolgt von Takt=0b1, Rest=0xf.

Im Anschluss werden die Flickerdaten, jeweils beginnend mit dem niederwertigsten Halbbyte, übertragen. Mit der steigenden Flanke des Takts beginnt ein neues Halbbyte.

Am Beispiel von 0F0487…:

Wert	Takt	2^0 (1)	2^1 (2)	2^2 (4)	2^3 (8)
F	1	1	1	1	1
0	0 1	1 0	1 0	1 0	1 0
	0	0	0	0	0
4	1	0	0	1	0
	0	0	0	1	0
0	1	1	0	0	0
	0	1	0	0	0
7	1	1	1	1	0
	0	1	1	1	0
8	1	0	0	0	1
	0	0	0	0	1

TAN-Generator

Zur TAN-Erzeugung wird die Bankkarte in den TAN-Generator gesteckt und mit der Funktionstaste „F“ das Einlesen gestartet. Der Flickercode wird über die Fototransistoren auf der Rückseite eingelesen.

Die Tests erfolgten größtenteils mit einem REINER SCT tanJack® optic SR.

Die Schritte im Einzelnen:

a. Initialisierung

Nach dem Einstecken der Bankkarte ist mit einem Logic Analyzer Saleae Logic folgendes an Kontakten der Smartcard zu beobachten:

- Spannungsversorgung via VCC startet
- I/O wird auf HIGH gesetzt
- das CLK-Signal startet
- RST wird auf HIGH gesetzt

Abb. 2: Kontakte [2]

Die Kommunikation im Weiteren auf der I/O-Leitung zwischen Terminal und Karte wurde mittels Bus Pirate v2go mitgeschnitten.

Testaufbau:

Signalleitung	Bus Pirate Pin
GND	GND
CLK	AUX
I/O	MISO

Die Signalleitungen VCC, RST, CLK, GND, I/O müssen zwischen Terminal-interface (im Bild unter Zuhilfenahme eines passenden dünnen Platine) und der Chipkarte 1:1 verbunden werden. Die Signalleitungen GND, CLK und I/O werden des weiteren dem Bus Pirate zugeführt:

Folgende Firmwareversion des Bus Pirate kam zum Einsatz:

```
HiZ>i
Bus Pirate v3.a
Firmware v6.1 r1676  Bootloader v4.1
DEVID:0x0447 REVID:0x3042 (24FJ64GA002 B4)
http://dangerousprototypes.com
HiZ>
```

Parametrisierung des Bus Pirate [3]:

Um die Kommunikation mitschneiden zu können muss Anfangs muss die vom TAN-Generator der Karte vorgegebene Taktfrequenz ermittelt werden. Zu diesem Zweck bietet der Bus Pirate eine Frequenzmessung auf dem AUX PIN an; welche mit dem Befehl „f“ aufgerufen wird. Es ist empfehlenswert, diesen Vorgang mehrmals zu wiederholen. Die Messung ist nur nach dem Einführen der Karte in den Generator und während der Kommunikation mit der Karte möglich. Der Bus Pirate benötigt ein über eine Sekunde dauerhaft anliegendes Takt-Signal.

```
HiZ>ffffff
AUX Frequency:  autorange 0 Hz
AUX Frequency:  autorange 392,394 Hz
AUX Frequency: 450,304 Hz
AUX Frequency: 1,644,800 Hz
AUX Frequency:  autorange 392,394 Hz
AUX Frequency:  autorange 0 Hz
HiZ>
```

Mit Hilfe der Taktrate kann der Wert des BRG errechnet werden (vgl. Berechnung BRG):

```
(4000000/1644800*372)-1 = 903,66926 => 904
```

Damit kann nun der UART-Modus des Bus Pirate parametrisiert werden:

```
HiZ>m3
Set serial port speed: (bps)
1. 300
2. 1200
3. 2400
4. 4800
5. 9600
6. 19200
7. 38400
8. 57600
9. 115200
10. BRG raw value
(1)>10
Raw value for BRG (MIDI=127)

(34)>904
Data bits and parity:
1. 8, NONE *default
2. 8, EVEN
3. 8, ODD
4. 9, NONE
(1)>2
Stop bits:
1. 1 *default
2. 2
(1)>2
Receive polarity:
1. Idle 1 *default
2. Idle 0
(1)>
Select output type:
1. Open drain (H=Hi-Z, L=GND)
2. Normal (H=3.3V, L=GND)

(1)>
Ready

Mit "{" wird die Ausgabe gestartet:

UART>{
UART LIVE DISPLAY, } TO STOP
UART>
READ: -f 0x00
UART>

Damit gibt der Bus Pirate die einzelnen
Bytes in folgendem Format aus:

UART>
READ: 0x3B
UART>
READ: 0xFF
UART>
READ: 0x18
UART>
READ: 0x00
UART> ...
```

In den nachfolgenden Schritten werden die Byte mit Leerzeichen getrennt und, der Übersicht zuliebe, ohne 0x-Präfix dargestellt. Mit „T->C“ ist im Folgenden die Kommunikation vom TAN-Generator/Terminal zur Karte markiert, von der Karte zum Terminal mit „T<-C“.

b. ATR

Nach dem Anlegen der RST-Signals liefert die Karte den ATR (Answer to Reset) aus und der TAN-Generator überprüft, ob die ZKA TAN-Generatoranwendung angewählt werden kann. Im Anschluss beendet der Generator wieder die Kommunikation mit der Karte (alle Kontakte LOW) bis der Benutzer die Taste „TAN“ oder „F“ betätigt. Es startet die Initialisierungssequenz wie oben und der ATR wird erneut gesendet. In den Beispielen ist der Seriennummer des Chips im ATR mit „SS SS SS SS“ überschrieben.

T -> C	3B FF 18 00 FF 81 31 FE 45 65 63 11 08 43 02 50 00 10 SS SS SS SS 05 30 6C

Der ATR kann mit dem Script ATR_analysis aus den http://ludovic.rousseau.free.fr/softwares/pcsc-tools/ bequem aufbereitet werden:

```
 (pcsc-tools-1.4.18)$ ./ATR _ analysis  3B FF 18 00 FF 81 31 FE 45 65 63 11 08 43 02 50 00 10 SS SS
SS SS 05 30 6C
ATR: 3B FF 18 00 FF 81 31 FE 45 65 63 11 08 43 02 50 00 10 SS SS SS SS 05 30 6C
+ TS = 3B --> Direct Convention
+ T0 = FF, Y(1): 1111, K: 15 (historical bytes)
   TA(1) = 18 --> Fi=372, Di=12, 31 cycles/ETU
   129032 bits/s at 4 MHz, fMax for Fi = 5 MHz => 161290 bits/s
   TB(1) = 00 --> VPP is not electrically connected
   TC(1) = FF --> Extra guard time: 255 (special value)
   TD(1) = 81 --> Y(i+1) = 1000, Protocol T = 1
-----
TD(2) = 31 --> Y(i+1) = 0011, Protocol T = 1
-----
   TA(3) = FE --> IFSC: 254
   TB(3) = 45 --> Block Waiting Integer: 4 - Character Waiting Integer: 5
+ Historical bytes: 65 63 11 08 43 02 50 00 10 SS SS SS SS 05 30
   Category indicator byte: 65 (proprietary format)
+ TCK = 6C (correct checksum)
```

Demnach hat der Datenaustausch mit der Karte nach dem T=1 Protokoll zu erfolgen (auch ersichtlich an TCK. TCK wird nur bei T=1 gesendet).

T=0 vs. T=1:

- T=0: Byteorientiertes Protokoll. Jedem übertragenem Byte wird ein Parity-Bit angehängt. Ist die Parity nicht korrekt zieht der Empfänger die Leitung I/O auf HIGH und der Sender wiederholt anschließend das Byte.
- T=1: Blockorientiertes Protokoll. Die Kommandos und Antworten werden mit einem Header eingeleitet, am Stück übertragen und eine XOR-Prüfsumme angehängt. Bei einem Übertragungsfehler wird vom Empfänger der Block erneut angefordert (mit einer Steuernachricht).

Aufbau der Nachrichten:

- 1 Byte NAD (Node Address Byte)
- 1 Byte PCB (Protocol Control Byte)
  I Information Block
  R Receive Ready Block
  S Supervisory Block
- 1 Byte Länge 0..254 (FE)
- 0..254 Byte Daten
- 1 Byte XOR Prüfsumme ab und inkl. NAD
- NAD ist immer 0x00 (vorgesehen für zukünftige Verwendung)
- PCB siehe Tabellen unten

I Block (normale Datenübertragung)

b8	b7	b6	b5	b4 b3	b2	b1
0	Sende Sequenz (abwechselnd 0 und 1)	1: weiterer verketteter Datenblock folgt, sonst 0	0 (RFU)	0 (RFU) 0 (RFU)	0 (RFU)	0 (RFU)

R Block (Status)

b8	b7	b6	b5	b4 b3	b2	b1
1	0	0 (RFU)	Empfangs Sequenz (abwechselnd 0 und 1)	0 (RFU) 0 (RFU)	0	0: kein Fehler
					0	1: Checksummen- oder Parityfehler
					1: anderer Fehler	0

S Block (Protokollsteuerung)

b8	b7	b6	b5	b4 b3	b2	b1
1	1	0: Request, 1: Response	0	0 0	0	1: Resync
					1: Information Field Size (IFS)	0

IFS gibt die Puffergröße an, es wird hierbei zwischen Terminal und Karte unterschieden

- IFSD: Empfangspuffer des Terminals; Startwert 32 Byte
- IFSC: Empfangspuffer der Karte wird im ATR mitgeteilt (siehe oben: 0xFE, 254)

Quelle, abseits eingeschränkt verfügbarer ISO-Standards: http://www.panstruga.de/ct-api/spec/mktp2v09.ps.gz

Historical Bytes

Die „Historical Bytes“ (im Beispiel „65 63 11 08 43 02 50 00 10 SS SS SS SS 05 30“, siehe oben) des ATR haben bei SECCOS Karten folgende Bedeutung (Quelle)

Feld (1 Byte)	Wert
fix	65
fix	63
IC manufacturer ID Manufacturer's IC type ID	1 1 08
Manufacturer's ROM mask ID	43
Embedder country code	02
Embedder country code	50
Embedder national registration ID	00
Embedder national registration ID	10
Chip series number	SS
Chip series number	SS
Chip series number	SS
Chip series number	SS
Manufacturer OS Version (major version number)	05
Manufacturer OS Version (minor version number)	30

Die SECCOS-Version, hier „05.30“, ist von besonderer Bedeutung, da ab Version 6.0 der Ablauf geringfügig abweicht.

c. T=1 IFSD

Nachdem die Karte dem Terminal im ATR mitgeteilt hat, dass die Datenübertragung via T=1 erfolgt, wird ein S-Block Request geschickt um den IFSD zu erhöhen.

Das Terminal (T) teilt der Karte (C) per S-Block Request (110. ….) die Größe (FE = 254 byte) des Empfangspuffers IFSD (…. …1) mit:

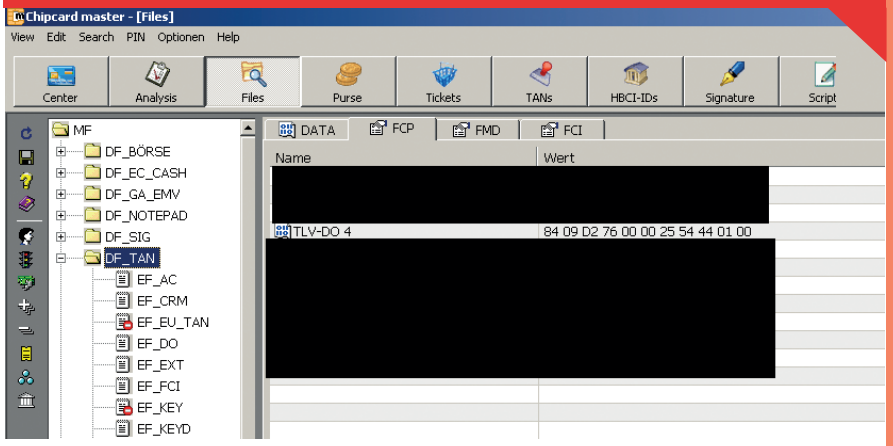
T->C 00 C1 01 FE 3E

Die Karte bestätigt dies per S-Block Response (111. ….).

T<-C 00 E1 01 FE 1E

Ab diesem Zeitpunkt folgen nur noch I Blöcke zwischen Terminal und Karte - die eigentliche Kommunikation. Innerhalb einer PCSC-Applikation können diese nachfolgenden APDUs direkt verwendet werden (ohne die vorderen drei T=1-Header- und das abschließende Prüfsummen-Byte)

Abb. 3: Chipcard Master



d. SELECT FILE

Im ersten Schritt wird mittels SELECT FILE die Datenstruktur des TAN-Generator ausgewählt. AID ermittelt in Chipcard Master:

<->	T=1 Header	APDU	T = 1 XOR
T->C	00 00 0E	00 A4 04 0C 09 D2 76 00 00 25 54 44 01 00	3B
T<-C	00 00 02	90 00	92

AID

Aufbau des AID (Application Identifier) aus RID (Registered Application Identifier) und PIX (Proprietary Application Identifier Extension) am Beispiel von „D2 76 00 00 25 54 44 01 00“:

```
AID: D2 76 00 00 25 54 44 01 00
RID: D2 76 00 00 25
   D: „national registration“
    2 76: ISO 3166 Country Code Deutschland
          00 00 25: ZKA?
PIX:              54 44 01 00: TAN Anwendung DF _ TAN
```

PIX 54440100 vgl. TAN Anwendung DF_TAN

e. GET PROCESSING OPTIONS

Nachfolgend wird das EMV Kommando GET PROCESSING OPTIONS (vgl. z.B. http://blog.saush.com/2006/09/08/getting-information-from-an-emv-chip-card/) abgesetzt; damit wird eine neue Transaktion gestartet.

GET PROCESSING OPTS ohne Parameter 83 00:

<->	T=1 Header	APDU	T = 1 XOR
T->C	00 40 08	80 A8 00 00 02 83 00 00	E1
T<-C	00 40 0E	77 0A 82 02 18 00 94 04 08 02 04 00 90 00	A5

Aufbau der Antwort (EMV TLV, http://www.emvlab.org/emvtags/all/):

```
77 0A 82 02 18 00 94 04 08 02 04 00
   82 Application Interchange Profile: 18 00
     Byte 1: bit 8: 1 = Initiate (not supported)
            bit 7: 1 = Offline static data authentication is supported
            bit 6: 1 = Standard offline dynamic data authentication is supported
            bit 5: 1 = Cardholder verification is supported
            bit 4: 1 = Terminal Risk Management is to be performed
            bit 3: 1 = Issuer Authentication is supported
            bit 2: 1 = Combined DDA/AC Generation is supported
            bit 1: RFU (0)
     Byte 2: RFU ("00")
             94 Application File Locator (AFL): 08 02 04 00
                   Indicates the location (SFI, range of records) of
                   the AEFs related to a given application
```

f. READ RECORD

Kartendaten einlesen, u.A. Kartennummer (wie Geldkarte, http://ftp.ccc.de/docs/cards/geldkarte.pdf)

<->	T=1 Header	APDU	T=1 XOR
T->C	00 00 05	00 B2 01 BC 00	0A
T<-C	00 00 1A	67 MM MM MM NN NN NN NN NN 9D 10 12 06 07 10 02 80 45 55 52 01 3F 00 01 90 00	..

Antwort der Karte gem. geldkarte.pdf:

```
67 MM MM MM NN NN NN NN NN 9D 10 12 06 07 10 02 80 45 55 52 01 3F 00 01
67: Branchenhauptschlüssel
   MM MM MM: „Kurz-BLZ" kartenausgebendes Institut
     NN NN NN NN NN: individuelle Kartennummer
            9D: Prüfziffer über 67 MM MM MM NN NN NN NN NN
                  10 12: Verfalldatum der ec-Karte
                        06 07 10: Aktivierungsdatum der ec-Karte
                              02 80: Ländercode
                                    45 55 52: Währungskennzeichen ‚EUR'
                                          01: Wertigkeit der Währung
                                                3F: Freier Zähler für die Schlüsselerzeugung
                                                      00 01
```

g. SEARCH RECORD IPB

Eine Art Volltextsuche in den Dateien der Karte. Es können keine Spezifikationen dieses Befehls gefunden werden. er ist vermutlich angelehnt an SEARCH RECORD (00 A2) aus der nicht offen gelegten Spezifikation ISO/IEC 7816-9, jedoch sind Informationen aus GSM-Standard ableitbar: CR #0019

Suche nach Tag „9F56" (issuer proprietary bitmap, „Bitfilter" (vgl. Optimised to Fail: Card Readers for Online Banking, Seite 30)

SECCOS vor 6.0

<->	T=1 Header	APDU	T=1 XOR
T->C	00 40 0F	00 A2 01 0F 09 04 00 CE 9F 56 00 00 00 FF 00	16
T<-C	00 40 1E	03 70 19 9F 55 01 38 9F 56 12 00 00 03 FF FF 80 00 00 00 00 00 00 00 00 00 08 00 00 90 00	07

SECCOS ab 6.0

<->	T=1 Header	APDU	T=1 XOR
T->C	00 40 0F	00 A2 01 0F 09 04 00 CE 9F 56 00 00 00 FF 00	16
T<-C	00 40 1E	03 70 19 9F 55 01 *00* 9F 56 12 00 00 03 FF FF 80 00 00 00 00 00 00 00 00 00 08 00 00 90 00	3F

Antwort der Karte gem. EMV TLV:

```
03 70 19 9F 55 01 38 9F 56 12 00 00 03 FF FF 80 00 00 00 00 00 00 00 00 08 00 00
   9F 55 01: issuer authentication flag (vermutlich ebenfalls Teil der Speicherstelle?), Länge
     38: Wert (00 bei SECCOS ab Version 6.0)
           9F 56 12: issuer proprietary bitmap, Länge
                  00 00 03 FF FF 80 00 00 00 00 00 00 00 00 08 00 00: Wert
```

h. SEARCH RECORD CDOL (SECCOS ab 6.0)

b SECCOS Version 06.00 setzt der TAN-Generator einen weiteren SEARCH RECORD-Aufruf für die Suche nach „8C“ ab. Das Feld „8C“ wird im EMV Standard CDOL1 bezeichnet und legt normalerweise fest, welche Daten dem später folgendem GENERATE AC Kommando übergeben werden sollen.

<->	T=1 Header	APDU	T=1 XOR
T->C	00 .. 0E	00 A2 01 0F 08 04 00 CE 8C 00 00 00 FF 00	..
T<-C	00 .. 33	02 70 2E 8C 21 9F 02 06 9F 03 06 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 9F 4C 08 9F 45 02 9F 33 03 9F 1A 02 9F 35 01 8D 09 91 10 8A 02 95 05 9F 37 04 90 00	..

Die Antwort enthalt sowohl CDOL1 als auch CDOL2 (letzteres wird üblicherweise für einen zweiten GENERATE AC Aufruf verwendet, dieser entfällt jedoch bei der TAN-Generierung). Aufgeschlüsselt gemäß http://www.emvlab.org/emvtags/all/ ergibt sich folgende Aufstellung. In der Darstellung findet sich vor dem Bindestrich das Feldkennzeichen (Tag), dahinter die Längenangabe.

```
CDOL1 8C 21 9F02-06 Amount, Authorised (Numeric)     1A 29 37 0D 90 46
   9F03-06 Amount, Other (Numeric)                    F7 A6 6C F7 62 5C
   95-05   Terminal Verification Results              53 25 51 78 4E
   5F2A-02 Transaction Currency Code                  5E 09
   9A-03   Transaction Date                           64 00 00
   9C-01   Transaction Type                           00
   9F37-04 Unpredictable Number                       00 00 00 00
   9F4C-08 ICC Dynamic Number                         00 00 00 00 00 00 00 00
   9F45-02 Data Authentication Code                   00 00
   9F33-03 Terminal Capabilities                      00 00 00
   9F1A-02 Terminal Country Code                      00 00
   9F35-01 Terminal Type                              00
CDOL2 8D 09 91-10   Issuer Authentication Data
   8A-02   Authorisation Response Code
   95-05   Terminal Verification Results
   9F37-04 Unpredictable Number
```

Den Beobachtungen nach ist nur die Summe der Längenangaben (hier 0x2B) für das spätere GENERATE AC der TAN-Generierung relevant.

i. VERIFY

Der nachfolgende Befehl VERIFY führt i.d.R. eine PIN-Verifikation durch. Bei deutschen Karten wird jedoch darauf verzichtet. Da einem späterem GENERATE AC ein VERIFY-Aufruf vorausgehen muss, wird stellvertretend die Kartennummer (NN NN NN NN NN) übergeben (siehe oben aus READ RECORD).

<->	T=1 Header	APDU	T=1 XOR
T->C	00 00 0D	80 A8 00 00 02 83 00 00	..
T<-C	00 00 02	90 00	92

j. HASH

Nach dem Scan und Bestätigung oder Eingabe der Transaktionsdaten (Kontonummer, Betrag) wird in zwei Schritten die eigentliche TAN ermittelt.

Die Transaktionsdaten werden mit ihren Maskenbezeichnungen vorweg aufbereitet und der der Karte übergeben, welche daraus einen Hash errechnet. Auch hier gibt es, ähnlich wie beim Flickercode eine Markierung ob es sich um ASCII (0xE1) und BCD (0xE0) handelt.

<->	T=1 Header	APDU	T=1 XOR
T->C	00 40 3C	00 2A 90 A0 36 90 00 80 81 31 E1 53 74 61 72 74 2D 43 6F 64 65 3A E0 87 11 00 03 E1 4B 6F 6E 74 6F 6E 75 6D 6D 65 72 E1 33 33 35 35 35 34 E1 42 65 74 72 61 67 E1 31 2C 32 33 B6 00	CF
T<-C	00 40 16	4B 4E 0A C4 13 E1 9D 4F A9 CA 17 7B 76 D4 AA 0B 5E 42 09 1C 90 00	28

Bezeichnung der „Nutz-Daten“ u.U. „VisDataSig“? (vgl. FinTS 3.0 Security Alternative Sicherheitsverfahren, B.5.1): Aufbau:

```
HASH                      1 a   S   t   a   r   t   -   C   o   d   e   : b   87 11 00 03 a   K   o   n   t
o   n   u   m   m   e   r   a   3   3   5   5   5   4 ..

00 2A 90 A0 36 90 00 80 81 31 E1 53 74 61 72 74 2D 43 6F 64 65 3A E0 87 11 00 03 E1 4B 6F 6E 74
6F 6E 75 6D 6D 65 72 E1 33 33 35 35 35 34 ..

  ..  a   B   e   t   r   a   g   a   1   ,   2   3   ¶x
  .. E1 42 65 74 72 61 67 E1 31 2C 32 33 B6 00
```

k. GENERATE AC

Erzeugung des Application Cryptograms. Bei SECCOS vor 6.0 werden 4 Byte des Hashes in eine ansonsten statische Struktur eingebettet übergeben. Ab SECCOS 6.0 wird der komplette Hashwert von 20 Byte verwendet und mit 23 Null-Bytes (0x00) auf insgesamt 0x2B (vgl. oben: SEARCH RECORD CDOL) Bytes aufgefüllt. In das Cryptogram fließen zudem karteninterne Daten/Schlüsselmaterial ein. Die Antwort der Karte ist nicht abhängig von der eingesetzten SECCOS Version.

SECCOS vor 6.0

<->	T=1 Header	APDU	T=1 XOR
T->C	00 00 31	80 AE 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00 09 99 00 00 00 00 4B 4E 0A C4 00 00 00 00 00 00 00 00 00 00 20 80 00 00 00 34 00	7B
T<-C	00 00 22	77 1E 9F 27 01 00 9F 36 02 01 02 9F 26 08 EC F5 0D 2C 1E AF 4E E2 9F 10 07 03 82 01 00 31 00 00 90 00	67

SECCOS ab 6.0

<->	T=1 Header	APDU	T=1 XOR
T->C	00 .. 31	80 AE 00 00 2B 4B 4E 0A C4 13 E1 9D 4F A9 CA 17 7B 76 D4 AA 0B 5E 42 09 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	..
T<-C	00 .. 22	77 1E 9F 27 01 00 9F 36 02 01 02 9F 26 08 EC F5 0D 2C 1E AF 4E E2 9F 10 07 03 82 01 00 31 00 00 90 00	..

Antwort der Karte (EMV TLV):

```
77 1E 9F 27 01 00 9F 36 02 01 02 9F 26 08 EC F5 0D 2C 1E AF 4E E2 9F 10 07 03 82 01 00 31 00 00
77 Response Message Template Format 2
   9F 27 Cryptogram Information Data (cid)
     00
            9F 36 Application Transaction Counter (ATC)
                  01 02
                        9F 26 Application Cryptogram (ac)
                              EC F5 0D 2C 1E AF 4E E2
                                    9F 10 Issuer Application Data
                                          03 82 01 00 31 00 00
```

l. Ableitung der TAN

Aus der Antwort der Karte wird nun final die TAN gebildet. Dazu werden die Nutzdaten der GENERATE_AC-Antworten aneinandergereit und mit dem Bitfilter (IPB) ver-AND-ed:

```
Daten: 000102ECF50D2C1EAF4EE203820100310000
IPB  : 000003FFFF80000000000000000000080000
```

```
Daten: 00000000000000010000001011101100111101010000110100101100000111101010111010011...
IPB  : 00000000000000000000001111111111111111111100000000000000000000000000000000...
AND  :                          1011101100111101010

Daten: ...10111000100000001110000010000000010000000000110001000000000000000000...
IPB  : ...00000000000000000000000000000000000000000000000010000000000000000000...
AND  : ...                                                                     0
```

Im Weiteren werden nur noch die Bit-Stellen aus den Daten verwendet, welche im Bit-Filter an korrespondierender Stellen mit 1 gekennzeichnet sind. Es entsteht eine komprimierte Darstellung.

```
Komprimiert: 10111011 0011 11010100
```

Im letzten Schritt werden die letzten acht Bit (LSBs) an den Anfang der Bitfolge gestellt (LEFT ROTATE 8) und die Zahl in Dezimalschreibweise dargestellt.

```
LEFT ROTATE um acht Bit:
                      11 11010100
           10111011
      =>   10111011    11 11010100
       => 251067
```

Damit ist die passende TAN 251067 zur Überweisung gefunden. Das Banksystem kann unter Anwendung der selben Eingangsdaten die TAN verifizieren.

Für Tests und die Verifikation von Vermutungen kam der Smart Card Detective zum Einsatz. Dieser unterstützt leider nur das Protokoll T=0; kann jedoch mit Quellcodemodfikationen zum „blinden“ Senden von Kartenantworten an den TAN-Generator verwendet werden. Mit Hilfe des SCD konnte der „Bit-Mix“ (IPB-anding, LEFT ROTATE, etc.) aus der Antwort auf GENERATE AC erkannt und rekonstruiert werden.

Links / Literatur

[1] siehe Webseite: http://wiki.ccc-ffm.de/projekte:tangenerator:start#t_1_ifsd

[2] Bild Quelle: Wikimedia Commons

[3] http://wiki.yobi.be/wiki/Bus_Pirate#7816-3_T.3D0_at_arbitrary_baudrate

Distance Measurement and Time Synchronization -

using Frequency Modulated Continuous Wave (FMCW) Radar

Niko Joram <niko.joram@tu-dresden.de>, Jens Wagner, Axel Strobel, Frank Ellinger

This article presents theory and design of a system for distance measurement and clock synchronization using radio waves with the frequency modulated continuous wave (FMCW) approach. The system consists of a custom analog front-end board and a Digilent Atlys Spartan 6 FPGA board. The analog front-end can transmit and receive frequency chirps in the 5.725 GHz to 5.875 GHz band. Test measurements in an indoor scenario show a maximum distance error of 1.15 m. A useful by-product of the distance calculation is the knowledge of the timing offset between two stations allowing precise wireless time synchronization in the single-digit nanosecond range.

Introduction

Nowadays, localization services are needed in many domains, be it the global positioning system GPS or its upcoming European counterpart GALILEO helping vehicle drivers to find their way, or the multiplicity of local positioning systems (LPS) with a variety of different applications. LPS can be found for example in smart factories to locate tools or allow their use only in particular areas, during guided tours in museums or navigation within malls; basically everywhere indoors. This is one of the main advantages in using LPS: the ability to operate in complex environments where GPS fails, like within buildings or areas outside which are densely covered with buildings or where other obstacles cause reflections of the used signals. Determining a 3D position using a LPS can always be traced back to measuring distances between base stations and mobile stations. Because of the fundamental nature of the distance measurement involved, we will present the design and implementation of modules using the frequency-modulated continuous wave (FMCW) radar for ranging, which will certainly find its applications also among hobbyists. The work presented in this article was done in the context of the European research project E-SPONDER [1]. The aim of the project is to provide information and communication support to first responders (e.g. fire fighters, paramedics) during large scale crisis events. Such events include forest fires, plane crashes or collapsing buildings during earthquakes. Localizing each first responder at all times contributes to the effective management of emergency situations. Since the nature of crises is many-sided, complex environments will certainly be encountered, which is a perfect match for a LPS. In the following section II, the basics of passive and active FMCW radar systems are described. The design of an analogue front-end and the digital signal processing back-end for an active FMCW system is presented in section III. Finally, some distance measurement results from the designed system are shown in section IV.

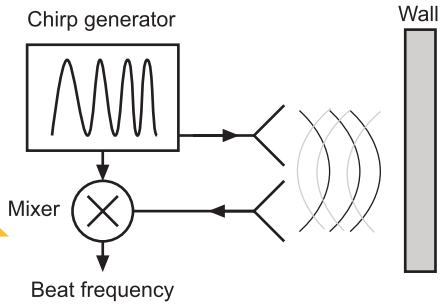


Abb. 1: passive-reflective

FMCW Basics

a. Passive Reflective Radar

The basic form of FMCW radar may be best known from the automotive sector, where it is used in electronic parking assistants or automatic interval control systems. Figure 1 shows a block diagram. A transceiver sends out a microwave frequency chirp, which propagates approximately with light speed. When the wave reaches an obstacle, it is being reflected and travels back to the transceiver. The signal now has a delay of two times the time-of-flight, measured from the point in time when it was transmitted.

The transceiver receives the reflected signal and mixes it with the transmitting chirp, which is not yet finished (the chirp duration is usually much larger than the expected time-of-flight). The result of the mixing process is a baseband frequency difference signal. In the ideal case of exactly one reflected path, this signal has a single frequency, the so-called beat frequency.

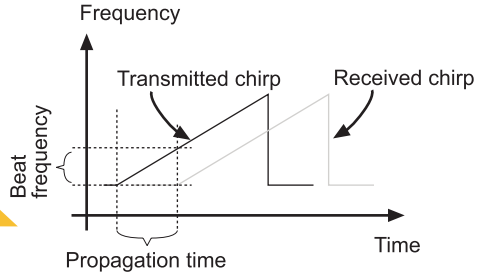


Abb. 2: chirps

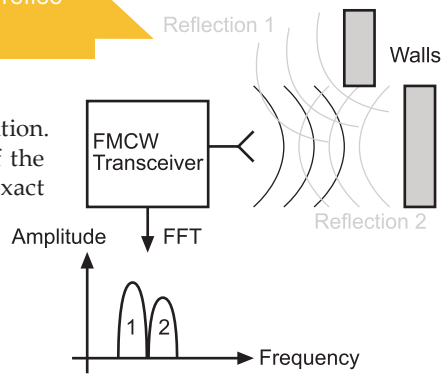
It carries the information of how long the signal has travelled until it was received again (i.e. two times the time-of-flight). Figure 2 shows how the propagation time of the chirp signal translates into a beat frequency. The beat frequency is usually detected by performing a Fourier Transform on the baseband signal and using a peak detector to detect the frequency with maximum amplitude.

As already mentioned, if the radio channel does not possess any multipath propagation, this signal has only one frequency component. Multiple signal reflections arriving at the receiver would appear as superimposed frequency components, usually with lower amplitude and higher frequency, because they take a longer path than the direct line of sight and thus arrive later at the receiver. As a result, multiple peaks with different amplitudes will appear in the spectrum.

To better understand the basics of FMCW radar systems, elementary mathematics are suf-

ficient. Table 1 summarizes the most important design equations. The propagation time of a signal can, of course, be calculated using the propagation speed (in the case of radio frequency (RF) signals, this is approximately the speed of light) and the distance from transmitter to receiver. In a RF FMCW system, the signal travels approximately 30 cm per nanosecond, whereas in an ultrasound system, it is only 30 cm per millisecond. The beat frequency can be calculated by multiplying the propagation time with the chirp gradient, which is the quotient of the chirp bandwidth and chirp duration. Another important parameter is the range resolution of the system. It is used for systems working in multipath environments (i.e. indoors). When we now perform the Fourier Transform, or Fast Fourier Transform (FFT) in practice, of the baseband signal containing frequencies of multiple reflections, we will ideally see a peak for each of the received reflections, as shown in Figure 3. Depending on the window function used for the FFT we get lobes with a certain width, rather than narrow spikes. When both lobes slide closer together, there is a point when they blend into each other and cannot be separated any more. Therefore, an important system parameter is the distance delta of two reflections which can still be separated. This is called the range resolution. This parameter is proportional to propagation speed and the inverse of the bandwidth of the chirp. In consequence it is expected that a wideband system will do a better job finding the exact direct line-of-sight reflection than a narrowband system. Consider as example automotive radar in the 76-81 GHz range, with a bandwidth of 5 GHz. Such a system would have a range resolution in the order of centimetres, while our narrowband system with 150 MHz bandwidth will have only 2 m.

Abb. 3: reflections

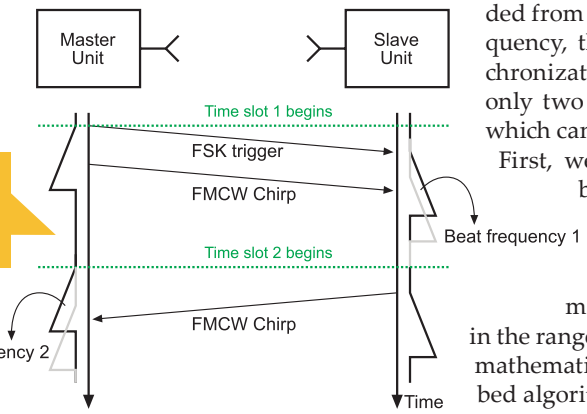


b. Active Reflector Radar

Until now, we assumed that the signal is reflected from objects such as walls. It is also possible to use two FMCW transceivers and determine the distance between them using the same principles. However, the main problem which directly arises is the synchronization of both units. To get the absolute propagation time and therefore real distance between two units, the chirp of the transmitting one has to start at the same time as the local chirp of the receiving unit.

The first idea of a solution which comes to mind is to use a cable between both units, which triggers the receiving unit when the transmission starts. The propagation time of the trigger signal along the cable is constant and can be removed from the final result. But using a synchronization cable in an otherwise wireless system seems somehow inflexible.

Abb. 4: synchronization



When transmitting the trigger signal wirelessly, e.g. via a FSK communications module, the propagation time is unknown again. Furthermore, in standard FSK transceiver chipsets there will always be uncertain processing delays until the signal is sent or a received signal is detected and decoded. Those delays are usually in the order of microseconds and cannot be neglected (remember that one nanosecond means that the RF signal has travelled 30 cm). Thus, we have to find a way to cancel out those delays when performing the range measurement.

A possibility is the use of an active reflector as a slave unit. The timing diagram is depicted in Figure 4. In the first time slot, the master unit sends out a trigger signal using FSK, and after an unknown delay it is detected by the slave unit. The only requirements in this step are that the delay is short enough, that the received chirp and the locally generated chirp in the slave unit still overlap to get a beat frequency, and that the resulting beat frequency is low enough to be processed by the circuitry. After sending out the trigger, the master sends a chirp signal. After detecting the trigger, the slave generates its own chirp to be mixed with the chirp from the master station. The slave then records the resulting beat frequency 1 and saves it (it contains the actual propagation delay and the synchronization delay). After master and slave units waited a fixed amount of time, the second time slot begins, but this time in the other direction: The slave unit now sends a chirp to the master unit, while the master unit generates a local chirp. The master then records and saves beat frequency 2. By subtracting the first recorded from the second recorded beat frequency, the contained uncertain synchronization delay cancels out, leaving only two times the propagation time, which can now be used for two things: First, we can calculate the distance between the units. Second, we know now the time difference between the clocks of both units, which allows a mathematical synchronization in the range of below 10 ns. An elaborate mathematical derivation of the described algorithm can be found in [2].

The major drawback of this algorithm is that both stations need to transmit and receive. Imagine a setup with several beacons acting as base stations and a number of mobile stations. Every mobile station would have to exchange chirps with every base station, which drastically decreases the measurement rate of the system. It would be beneficial, if the mobile station only receives signals from the base stations, calculating the distances and therefore its position on its own. This can be reached by employing the time difference of arrival (TdoA) or multilateration scheme, which is widely known from GPS. This scheme can also be implemented using the hardware described in this work; the interested reader is encouraged to devise an algorithm!

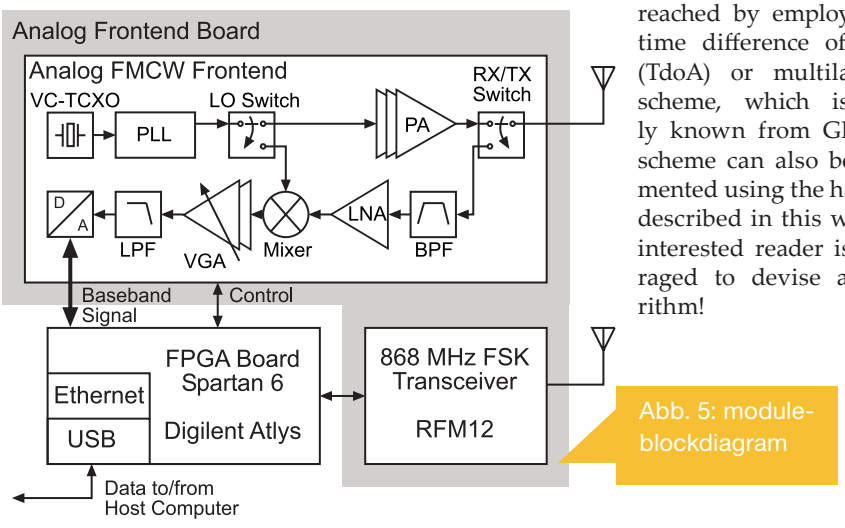


Abb. 5: module-blockdiagram

Hardware Design

a. A.Module Overview

From the considerations above, we can now identify which components we will need in the distance measurement system:

- An analogue front-end, which consists of a chirp generator, RF transmitter and receiver,
- a FSK transceiver for sending the trigger signal,
- a digital back-end for controlling the analogue part and the FSK transceiver and for interfacing with the outside world (e.g. a PC with USB or Ethernet)

A block diagram is presented in Figure 5; a photo of the whole module is shown in Figure 6.

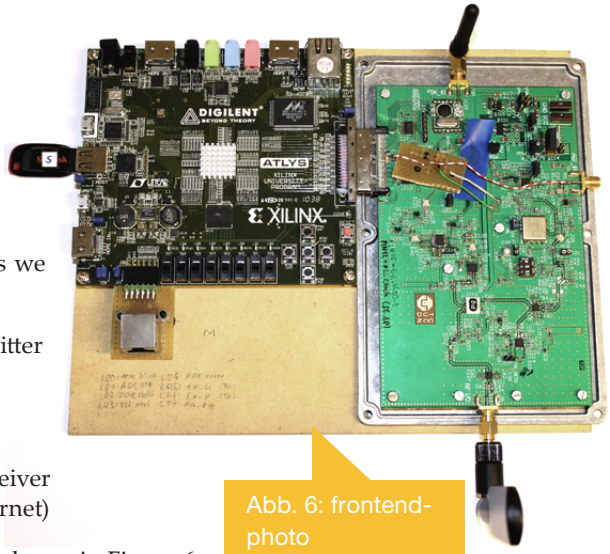


Abb. 6: frontend-photo

b. FSK Transceiver

For the FSK transceiver, we chose the low-cost and easy-to-use RFM12 module from HopeRF [3]. The first step was to perform measurements of the delay uncertainty of the module. The test bed consists of two RFM12 modules each connected to an AVR microcontroller. One module sends a certain data packet and generates a trigger pulse. The packet is received by the second station. Upon detection in the receiver, a second trigger pulse is generated. Both trigger pulses are visualized on an oscilloscope to determine the delay with the result, that the modules have a fixed delay between the triggers of roughly 3 ms and an additional uncertain delay of roughly 10 µs. Therefore, after subtracting the fixed delay, the FSK trigger signal arrives at the slave station with 10 µs uncertainty, which is low enough to ensure that the chirps overlap and that the beat frequency does not get too high to be processed by the used amplifiers.

c. Analogue Front-end

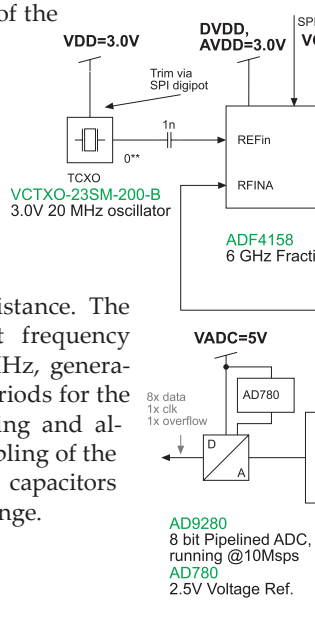
The analogue front-end of the system basically consists of a chirp generator, a transmitter and a receiver. A schematic is shown in Figure 7.

The heart of the chirp generator is the ADF4158 PLL chip [4]. It can work up to 6 GHz and includes a special counter to generate chirp waveforms. This counter increases the output frequency in pre-programmed steps with each count. It can be clocked internally by issuing an SPI command or externally using the TX-DATA pin as a trigger input. Using the external trigger is preferable, because our measurements showed that an uncertain delay time passes from issuing the SPI command until the chirp is starting. The PLL is clocked by a voltage-controlled temperature-compensated crystal oscillator (VC-TCXO) running at 20 MHz. It can be trimmed in a narrow frequency band of ±15 ppm by a control voltage, which is generated using a voltage divider with an SPI digital potentiometer. With this trimming possibility, the reference frequencies of other stations can be aligned to one master station, which eliminates errors in the system. It is advisable not to save money on this part, because the frequency drift of cheap oscillators can be substantial for the ranging accuracy. The specified stability of the used oscillator is ±2.5 ppm. The on-board oscillator on the Atlys Board was not suitable, because during the measurement cycle, which is in the range of 10 ms, the frequency had already drifted measurably. Its stability is specified with more than ±50 ppm.

The voltage-controlled oscillator (VCO) is driven by a control voltage generated by the PLL chip together with a filter. It operates in the band from 5.725 GHz to 5.875 GHz. The output signal of the VCO is split by a resistive divider, which provides 50 Ohms impedance on each of the 3 ports. One part of the signal is fed back to the PLL while the other part goes to a buffer amplifier, which compensates for the loss of the resistive power splitter.

After the LO buffer, a RF switch is used to select if the chirp signal goes to the mixer (in receive mode) or to the power amplifier (transmit mode). To always provide defined terminations of 50 Ohms on all ports, a DPDT switch is used, which always connects the unused path to a 50 Ohms resistor. The following power amplifier (PA) is an off-the-shelf WLAN type, having a maximum output power of 25 dBm. After the PA, another DPDT switch selects the connection of the antenna between transmit and receive path. The receive path consists of the standard parts, which is band filter, RF low noise amplifier (LNA), mixer, channel filter, variable gain amplifier (VGA), anti-aliasing low pass filter and analog-to-digital converter (ADC). It is worth noting, that the gain of the AD8330 baseband VGA can either be controlled from the FPGA using a digital potentiometer or it can be used in automatic gain control (AGC) mode. In AGC mode, the gain is set automatically during the measurement, such that the ADC is driven with maximum voltage levels. The ADC has 8 data bits and a 10 MHz clock, which is generated by the clock manager inside the FPGA.

The receiver can be considered a low intermediate frequency type. When the distance of two stations is low, it would mean getting a very low beat frequency in the Hz range. However, when processing frequencies near DC in the baseband, there are several problems arising. First, since the duration of the measurement cycle is in the order of milliseconds, only fractions of one period of the beat frequency would be generated, which would be problematic when calculating the Fourier Transform. Also, the building blocks either need the ability to be DC coupled or large blocking capacitors need to be used. To avoid those problems, the beat frequency is increased by deliberately delaying the generation of the local chirp by a constant time. Since the two-way ranging algorithm is used, this additional delay cancels out when calculating the distance. The minimum beat frequency was set to 1 MHz, generating enough periods for the digital processing and allowing the coupling of the amplifiers with capacitors in the 100 nF range.



d. Digital Back-end

The main task of the digital back-end is to process the sampled data from the ADC, calculate the FFT, estimate the peak in the spectrum and calculate the corresponding distance. Furthermore, it takes care of the control of all involved hardware. The digital design is implemented using Verilog HDL on the Digilent Atlys FPGA board, which contains a low-cost Spartan 6 LX45 FPGA (current price of the FPGA is around 50 EUR). A block diagram is depicted in Figure 8. The digital design consists of two major parts, the FMCW top-level, which interfaces with the analogue front-end, and the CPU top-level, which provides the user interface to the system. Also, there is a DDR memory controller created using Xilinx Memory Interface Generator (MIG), providing independent access ports to the external memory. Finally, a digital clock manager (using the PLL_ADV primitive), driven by the 20 MHz VC-TCXO, creates and distributes clocks for all the building blocks.

The FMCW top-level contains the FMCW signal processing blocks. A sampler fetches the values from the ADC and saves them to DDR memory, so the signal can be evaluated later in MATLAB. From the sampler, the signal goes to a windowing block, which applies a Hamming window before calculating the FFT. The Hamming window helps to suppress side lobes in the spectrum, which would result from rectangular or no windowing. The Hamming window is implemented using a look-up table with precalculated 8 bit values from MATLAB. The following FFT block is implemented using the Xilinx Fast Fourier Transform IP with a transform length of 65536 (or 64K) and Radix-2 Burst I/O architecture. The arithmetic is scaled fixed-point with 8 bit phase factor. Since frequency directly translates into distance, it is beneficial for the accuracy of the system to make the transform length large. With a 64K transform and 10 MHz sampling frequency, we get 153 Hz per frequency bin. Using the 150 MHz bandwidth of the 5.8 GHz band and 2.5 ms chirp duration, we get about 76 cm per frequency bin. At first glance this value seems rather large, but it can be drastically increased by doing a polynomial interpolation using the detected peak and its neighbours and then calculating the apex. Another possibility would be to reduce the chirp duration, but this again would lead to less signal portion and more zero padding in the FFT. The calculated spectrum and detected peak bin are saved to the DDR memory for further evaluation by the CPU (e.g. polynomial interpolation).

Furthermore, there are several separate SPI masters, which are connected to the PLL, the RFM12 FSK transceiver or the digital potentiometers for trimming the VGA or VC-TCXO.

The heart of the FMCW top-level is the user interface. It can be considered as a batch processor. It contains two FIFOs and a state machine for processing the FIFO contents and controlling the peripherals. The RX FIFO receives a series of commands (i.e. a program or job) from the Wishbone bus. This program can, for example, be the two-way ranging algorithm (send a FSK trigger, send chirp, receive chirp, detect peak). After the complete program is stored, it is executed as a batch job. Each peripheral in the FMCW top-level has a set of commands, and several of those commands can be compiled into jobs. Other than running those jobs on a conventional CPU, this approach has the advantage that the execution times are predictable and always equal to those of the last run. Of course, special attention also has to be paid to the timing of the peripherals. In the TX FIFO, the peripherals can store the results of the operations, if any. This can for example be the peak frequency bin from the peak detector.

The second major part is the CPU top-level. It consists of an AVR CPU core with program and data RAM and several peripherals, which are interconnected using a Wishbone bus. The AVR core with the Wishbone interface was adapted from the BitHound project [5]. It is running on a 50 MHz clock. The task of the CPU is to provide a high-level programming interface and user interface for the system to test ranging and signal processing algorithms. The AVR core is compatible to the ATMega103, thus the avr-gcc tool chain together with AVR Studio can be used to write and compile C programs for the CPU. To merge the compiled ELF binary for the CPU with the BIT file from Xilinx ISE, the data2mem tool from Xilinx is used. This has the advantage of allowing a quick test of the firmware, because the time-consuming synthesis of the digital design needs to run only once.

The connection between the FMCW top-level and the CPU is also done using the Wishbone bus. There are several registers to write and read the FIFOs and get the status of the FMCW peripherals, which are mapped to the I/O address space of the CPU and can therefore be accessed easily. It is also possible to use the CPU to transmit the saved samples or FFT result from DDR memory via UART [6] or Ethernet [7] to a host system (e.g. to a PC using the USB-to-UART bridge on the Atlys board, some examples will be shown in the next section). Using a simple Wishbone-to-MIG translator (translating between Wishbone bus cycles and the FIFO interface of the MIG core), it is possible to read and write to DDR memory. The address and data registers for accessing the DDR memory are memory-mapped to the AVR's address space.

The GPIO [8] connects to the LEDs, switches and Pmod expansion connector on the Atlys board and is used for status display or to connect a LC display.

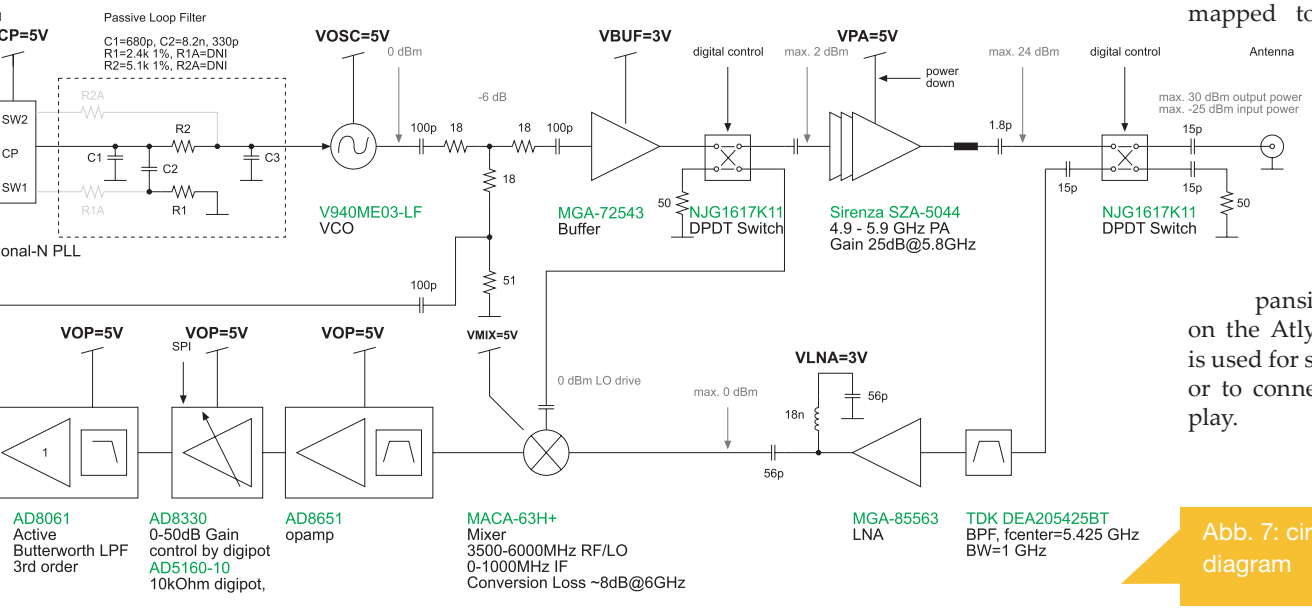


Abb. 7: circuit-diagram

Assembly and Measurement

a. Printed Circuit Board

A few words on PCB design are necessary to be successful with a RF project of this type. For our design, we used a 2 layer FR-4 board with 0.5 mm thickness to save costs for a first prototype. The analogue board has a size of 160x100 mm². The small thickness allows 50 Ohms transmission lines to be smaller in width than with the standard 1.55 mm boards. The transmission lines were designed as a mixture of coplanar and microstrip which means the actual line is surrounded with a ground plane on the same layer and a ground plane on the lower layer beneath the line. Along the line, vias were used to connect the top and bottom ground planes. Figure 9 shows a detail of the used lines. The following dimensions were determined by simulating the lines with SONNET [9]: 0.8 mm line width, 0.4 mm clearance to ground plane on top layer. Please note that the ground plane below a transmission line on the bottom layer must not be intersected, because it also carries part of the RF return current. With a 2 layer board, this may occasionally be difficult, which is the reason why most RF mixed-signal designs use 4 layer boards. When using a 4 layer board, it is important to use a special layer stacking consisting of two 0.5 mm FR-4 boards, which are glued together using one prepreg layer to get defined distances between signal and ground layer.It is also recommended to use SMA board edge connectors instead of the trough-hole variants, because they provide better return loss, especially at frequencies as high as 5.8 GHz. Our measurements show a good return loss of 10 dB at 6 GHz using two SMA edge connectors with a 5 cm transmission line (as described above) as a connection.Furthermore, it has to be noted that the RF components on the boards tend to influence each other, when in close proximity (e.g. less than 1 m), which leads to increased spurious emissions. To avoid such problems, it is recommended to shield the RF components such as PLL, power amplifier and LNA using RF shielding boxes.

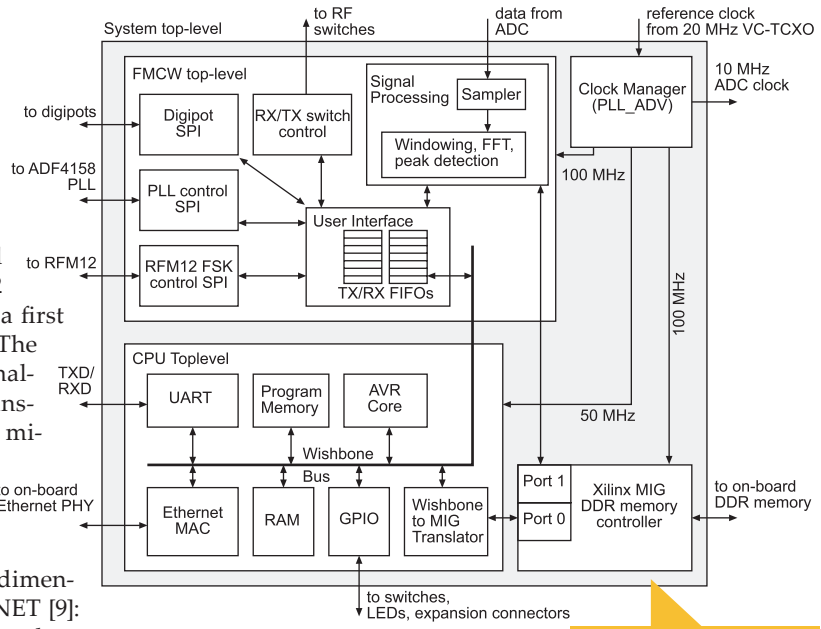


Abb. 8: digital part

b. Ranging Results

To conduct first tests of the system, we set it up in two scenarios: indoors in a corridor and outdoors on a lawn. This article will focus on the results from the indoor scenario. The corridor has a length of 24.2 m, width of 3.2 m and height of 3.4 m. We used two stations and performed ranging with different distances. To directly save the baseband signal and spectrum during the measurements, we connected a Laptop to each of the stations. A photo of the indoor setup is shown in Figure 10. One of the stations is moved in the distance range from 1 m to 21 m. For each distance, 20 measurements are performed and the results recorded. The measurement result from the system is checked against the result from a laser distance meter with accuracy in the millimetre range.

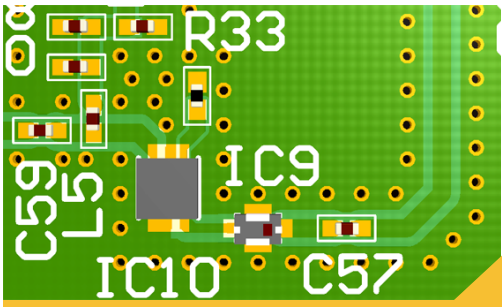


Abb. 9: transmission line

For all measurements, we use a chirp bandwidth of 150 MHz and chirp duration of 2.5 ms. Figure 11 shows a recorded time domain signal and corresponding spectrum from one measurement at 1 m distance. On the horizontal axis, the so-called pseudo distance is plotted. This is the direct translation of the measured beat frequency into distance using the last equation of Table 1. The line-of-sight path can clearly be seen as the strongest peak. However, there are also weaker side lobes, which represent signal reflections, which probably come from the floor or ceiling.

The distance error of the whole measurement series is presented in Figure 12. It is defined as the difference between the real distance and the mean distance measured by the system during the 20 single measurements. The first eye-catching thing is that the error is constantly above approximately 3 m, which means that the system has an inherent distance offset. This constant offset was also observed during the outdoor measurement and can be considered as a system constant. The offset is probably caused by the phase responses of the RF front-end components from the antenna switch to the mixer. To measure the offset, the two boards are connected directly together at their RF ports and the distance is measured. For our system, the offset seems slightly temperature dependent. It amounts to 3.07 m at room temperature, while it can go down to 2.92 m at 6°C.

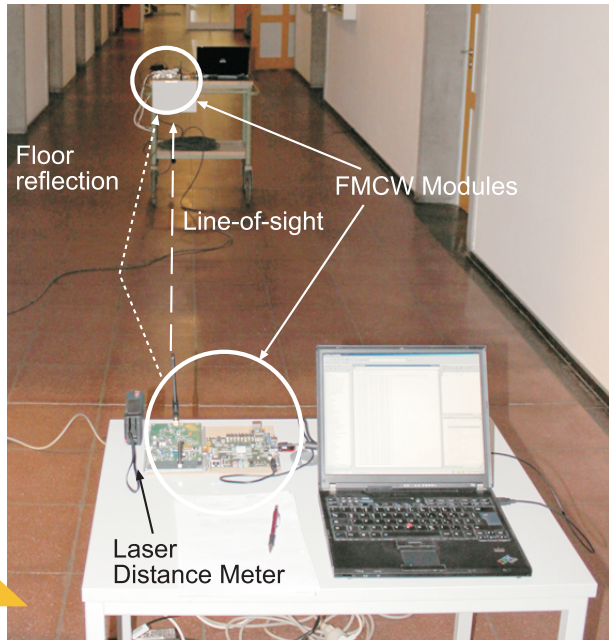


Abb. 10: meas setup

Subtracting the offset, the maximum distance error for the indoor scenario amounts to 1.15 m, which translates to a synchronization uncertainty of only 3.8 ns. The minimum distance error is also worth investigating, as it amounts to 20 cm. This can be explained by multipath propagation, since the side lobes from the reflections are superimposed on the line-of-sight lobe, which eventually shifts the peak towards higher frequencies.

Conclusion and Outlook

In this article we demonstrated that wireless distance measurement and precise time synchronization in the nanosecond regime can be accomplished by using off-the-shelf analogue components together with digital signal processing using a low-end FPGA. The basics of FMCW systems were discussed in detail and design strategies and decisions were explained, so that the interested engineer can reproduce the design.

As an outlook, there are several things which can still be investigated. One would be the design of a FMCW system using ultrasound components. Better accuracy can be expected, because the propagation speed of sound is much lower than that of electromagnetic waves. With bandwidths of several kHz of typical ultrasound transmitters and receivers, a range resolution in reflective environments in the order of 10 to 20 cm is to be expected.

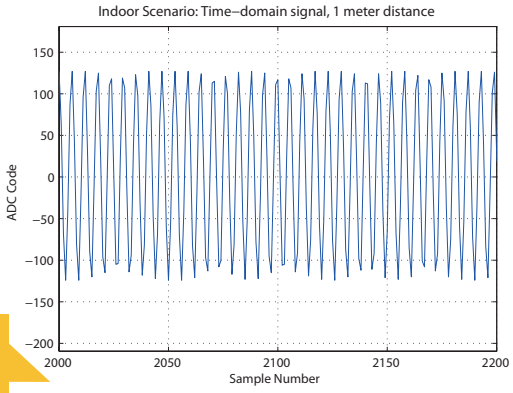


Abb. 11: timedomain indoor

Another point, which is important, is the detection of the line-of-sight signal in through-wall measurements. In those environments, a simple peak detector would give wrong results in many cases. Especially for systems operating with RF signals, through-wall measurement capability would mean a definitive advantage over classical ultrasound systems.

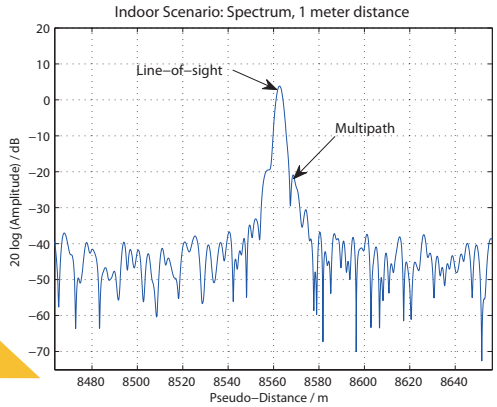


Abb. 12: spectrum indoor

Finally, for the design of a complete positioning system, it is necessary to think about data fusion to enhance the significance of the measurement result. There are several possibilities, e.g. the use of multi-band systems to get multiple uncorrelated positioning data from different frequency bands or the use of an inertial platform, which uses a compass together with accelerometers and atmospheric pressure sensors to provide tracking support to the wireless position measurement.

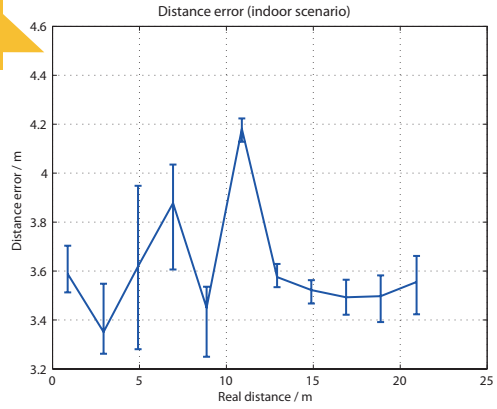


Abb. 13: disterr indoor

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°242411 (E-SPONDER).

Furthermore, the authors would like to thank the people in the mikrocontroller.net community for providing many ideas and support.

References

[1] EU Project E-SPONDER: http://www.e-sponder.eu
N. Joram, J. Wagner, A. Strobel and F. Ellinger, '5.8 GHz Demonstration System for Evaluation of FMCW Ranging', 9th Workshop on Positioning Navigation and Communication (WPNC), Dresden, Germany, 15-16 March, 2012

[2] RFM12 FSK module: http://www.hoperf.com/upload/rf/RFM12.pdf

[3] Analog Devices ADF4158 PLL: http://www.analog.com/static/imported-files/data_sheets/ADF4158.pdf

[4] BitHound project: http://www.bastli.ethz.ch/index.php?page=BitHoundEn

[5] UART core: http://opencores.org/project,uart16550

[6] Ethernet MAC: http://opencores.org/project,ethmac

[7] GPIO core: http://opencores.org/project,gpio

[8] SONNET EM simulation software: http://www.sonnetsoftware.com/products/lite/

Marktplatz / Neuigkeiten

Die Ecke für Neuigkeiten und verschiedene Produkte

Ankündigung: Zweiter Mikrocontroller.net Artikelwettbewerb

Kennst du dich mit einem Thema besonders gut aus? Hast du ein interessantes Projekt das du schon lange mal vorstellen wolltest? Dann kannst du bis zum **1.3.2013** die Gelegenheit nutzen einen Artikel zu schreiben und einen von 33 Preisen im Wert von bis zu 499 € zu gewinnen. Mikrocontroller.net veranstaltet mit dem Embedded Projects Journal zum zweiten Mal einen Artikelwettbewerb (der 1. Wettbewerb fand 2008 statt), bei dem es darum geht Artikel zu Elektronik- oder Embedded Systems-Themen auf Mikrocontroller.net zu veröffentlichen, von denen die besten prämiert und im Embedded Projects Journal veröffentlicht werden.

b. Teilnahme

So kannst du teilnehmen: Auf Mikrocontroller.net anmelden (wichtig, damit man sieht wer welchen Artikel geschrieben hat!); Artikel in der Mikrocontroller.net Artikelsammlung erstellen (Anleitung); "[[Kategorie:Wettbewerb]]" zum Text hinzufügen.

c. Regeln

Der Artikel muss vollständig und in sich abgeschlossen sein. Kein Lexikonartikel, keine Aufzählungen oder Linksammlungen. Der Artikel darf nicht schon irgendwo in ähnlicher Form veröffentlicht worden sein. Text und Bilder des Artikels müssen wie alle Inhalte der Artikelsammlung unter der Creative-Commons-Lizenz freigegeben werden, Code unter einer Open Source Lizenz.Der Artikel muss bis zum 1.3.2013 fertig in der Mikrocontroller.net Artikelsammlung eingestellt sein.

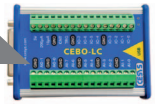
a. Die Preise

Diesmal gibt es insgesamt 33 Preise im Gesamtwert von 2222 €.

Platz 1: Digital-Speicher-Oszilloskop UTD2062CM, 1GS/s, im Wert von 499 €

reichelt elektronik

Platz 2-4: USB Messlabor im Wert von 299 €



CESYS

Platz 5: True-RMS-Multimeter Voltcraft VC880 im Wert von 179 €

CONRAD

Platz 6-8: GNUBLIN Embedded Linux Board im Wert von 49 €

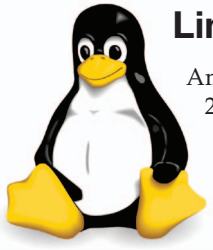
embedded projects GmbH
HARDWARE FOR PROJECTS

Platz 9-33: Platinengutscheine von LEITON im Wert von 20 €

LEITON PRINTED CIRCUITS

d. Bewertung

Die Bewertung der Artikel wird von einer Jury bestehend aus den Betreibern von Mikrocontroller.net und dem Embedded Projects Journal, sowie Moderatoren des Mikrocontroller.net-Forums vorgenommen. Das Hauptkriterium ist: wie interessant und nützlich ist der Artikel für den Leser; zum Beispiel ist ein Artikel der ein allgemeines Thema erklärt (z.B. ein IC oder eine Programmiertechnik) in der Regel für mehr Leute nützlich als die Beschreibung eines sehr speziellen Projektes. Aber natürlich spielen auch Originalität, Form und Umfang des Artikels eine Rolle, und ggf. Codebeispiele oder Programmcode des Projektes.



Linux-Infotag Augsburg: Call for Participation

Am Samstag, den 23. März 2013 findet zum 12. Mal der Augsburger Linux-Infotag in den Räumen der Hochschule Augsburg statt. Unter dem Motto **„Linux überall"** soll die Veranstaltung Neulingen einen Einblick in die Anwendungsmöglichkeiten von Open-Source-Software geben und alten Hasen neueste Entwicklungen präsentieren. Organisiert wird der Linux-Infotag seit 2001 von der Linux User Group Augsburg (LUGA) e.V. und zählt zu den ältesten und größten Linux-Events im süddeutschen Raum.

Das Thema „Linux überall" spannt den Bogen zwischen den Aspekten „Linux auf einer Vielzahl unterschiedlicher Geräte" und „Linux in verschiedenen Anwendungsbereichen". Linux wird mittlerweile immer häufiger als Betriebssystem für Handys, Tablets und E-Book-Leser ebenso verwendet. Außerdem werden Linux-Systeme unterwegs und an verschiedenen Orten genutzt. Die eigenen Dateien liegen dabei entweder zentral auf einem Server („in der Cloud") oder auf dem Gerät. In beiden Fällen sollen die Daten vor Verlust (Backup) und vor den Augen Dritter geschützt werden.

Der Linux-Infotag soll diese beiden Aspekte näher beleuchten. Den Einsteigern soll dabei ein allgemeiner Einblick und Entwicklern sowie langjährigen Linux-Nutzern ein Überblick über die neuesten Entwicklungen gegeben werden. Dazu suchen wir Vorträge und Workshops, sowie Projekte und Gruppen, die sich mit einem Stand präsentieren wollen. Neben Einsteiger-Vorträgen und -Workshops suchen wir vor allem – aber nicht ausschließlich – Beiträge zu folgenden Themengebieten:

- Linux auf mobilen Geräten: Android, Embedded Linux etc.
- Schutz der eigenen Daten: Backups, dezentrale Dateisysteme, Verschlüsselung, Cloud-Lösungen wie ownCloud
- Linux und Bildung: Freie Software in Schulen und Universitäten

Weiter sind auch Vorträge und Workshops rund um Linux, Open-Source-Software und digitale Gesellschaft (Wikipedia, Bitcoin, Netzzensur...) willkommen.

Bitte schickt uns Eure Vorschläge bis spätestens Sonntag, 13. Januar 2013, an lit-vortraege@luga.de (für Vorträge und Workshops) bzw. an lit-staende@luga.de (für Projektstände).

Interesse an einer Anzeige?

info@embedded-projects.net

embedded - projects.net
JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

Bitte frei machen

embedded projects GmbH
Holzbachstraße 4
D - 86152 Augsburg

Name / Firma

Straße / Hausnummer

PLZ / Ort

Email / Telefon / Fax

◯ Ich möchte jede zukünftige Ausgabe erhalten

◯ Wir möchten als Hochschule / Ausbildungsbetrieb jede weitere Ausgabe bekommen. Bitte gewünschte Anzahl der Hefte pro Ausgabe ankreuzen. 5 10

◯ Ich möchten im embedded projects Journal werben oder eine Stellenanzeige aufgeben. Bitte schicken Sie mir Infomaterial, Preisliste etc. zu.

Werdet aktiv!

Das Motto: Von der Community für die Community !

Das Magazin ist ein Open Source Projekt.

Falls Du Lust hast, Dich an der Zeitschrift durch einen Beitrag zu beteiligen, würden wir uns darüber sehr freuen. Schreibe deine Idee an:

journal@embedded-projects.net

Regelmäßig

Die Zeitschrift wird über mehrere Kanäle verteilt. Der erste Kanal ist der Download als PDF - Datei. Alle Ausgaben sind auf der Internetseite [1] verfügbar. Diejenigen, die lieber eine Papierversion erhalten möchten, können den zweiten Kanal wählen. Man kann sich dort auf einer Internetseite [2] in eine Liste für die gesponserten Abos oder ein Spendenabo eintragen. Beim Erscheinen einer neuen Ausgabe wird dank Sponsorengeldern an jeden auf der Liste eine Ausgabe des aktuellen Journal versendet. Falls man den Versandtermin verpasst hat, kann man das Heft auch über einen Online - Shop [2] beziehen.

[1] Internetseite (Anmeldeformular gesponserte Abos): http://journal.embedded-projects.net

[2] Online - Shop für Journal:
http://www.embedded-projects.net

Sponsoren gesucht!

Damit wir weiterhin diese Zeitschrift für jeden frei bereitstellen können, suchen wir dringend Sponsoren für Werbe- und Stellenanzeigen. Bei Interesse meldet Euch bitte unter folgender Telefonnummer: 0821 / 2795990 oder sendet eine E-Mail an die oben genannte Adresse.

Impressum

embedded projects GmbH
Holzbachstraße 4
D-86152 Augsburg
Telefon: +49(0)821 / 279599-0
Telefax: +49(0)821 / 279599-20

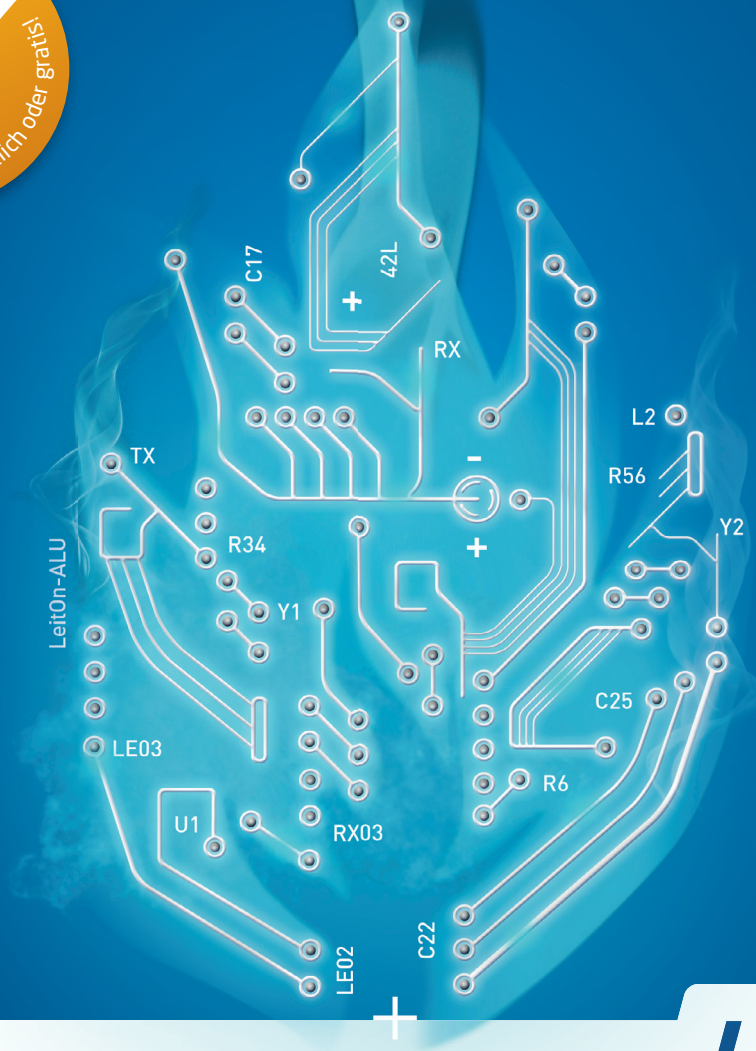
Veröffentlichung: 4x / Jahr
Ausgabenformat: PDF / Print
Auflagen Print: 2500 Stk.
Einzelverkaufspreis: 1 €

Layout / Satz: EP
Druck: flyeralarm GmbH
Titelfoto: Claudia Sauter

GERADE WENN'S MAL HEISS HERGEHT.

LEITERPLATTEN AUS ALUMINIUM ONLINE BESTELLEN.

ab 12 h Leiterplatten-Express! Pünktlich oder gratis!



www.H1QN.de

LEITON
RECHNEN SIE MIT BESTEM SERVICE

Vorsicht, heiß! Starke Hitzeeinwirkung ist eine echte Herausforderung – besonders in der Hochleistungs-LED-Technik oder wenn wichtige Leistungsbauteile permanent hohen Temperaturen ausgesetzt sind. Aluminium-Platinen von LeitOn überzeugen durch **hohe Hitzeresistenz** sowie einen Wärmeleitwert von bis zu 3,0 W/mK in der Premiumvariante „Polytherm TC-Lam 3.0". Die vielen Vorteile liegen auf der Hand: **Kostenreduktion** durch höhere Lebensdauer und Zuverlässigkeit, **Platzersparnis** dank Integration der aufwendigen Kühlmechanismen sowie **Performancesteigerung** durch höhere Leistungsdichte der Anwendung. Sie möchten mehr darüber wissen? Wir bieten persönliche Beratung am Telefon und einen kompetenten Außendienst. Sie können bei LeitOn immer mit dem besten Service rechnen.

www.leiton.de | **kontakt@leiton.de** | Info-Hotline **+49 (0)30 701 73 49 0**