MAI/JUNI 2025 | ELEKTORMAGAZINE.DE

€ 9,95
CHF 18,00

elektor
MEDIA & LEARNING

DIGITALE BONUS AUSGABE

609B

SEIT 1961

IM FOKUS

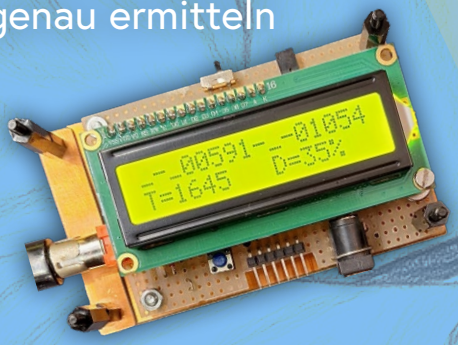
Prüf- und Messtechnik

Oszilloskop/ Multimeter/ Generator

Fnirsi 2C53T mit zwei Kanälen und 50 MHz Bandbreite

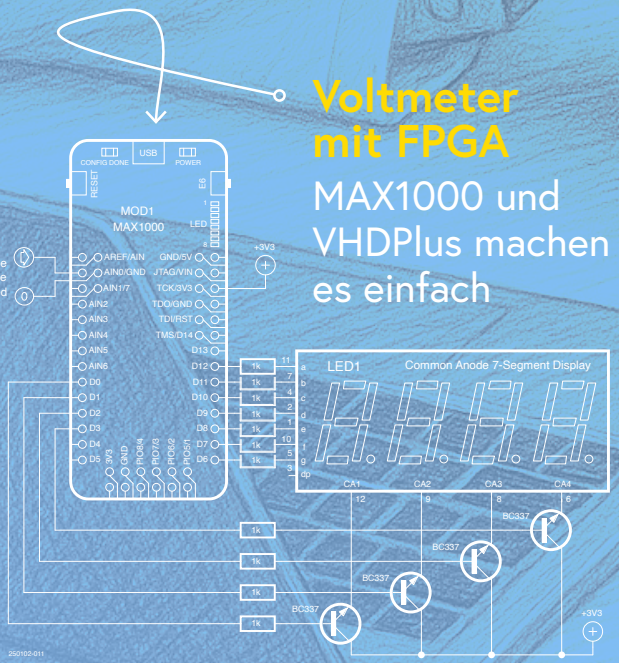
PWM-Messung mit einem PIC

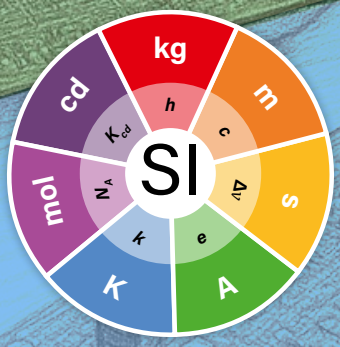
Timing eines Rechtecksignals genau ermitteln



Voltmeter mit FPGA

MAX1000 und VHDPlus machen es einfach



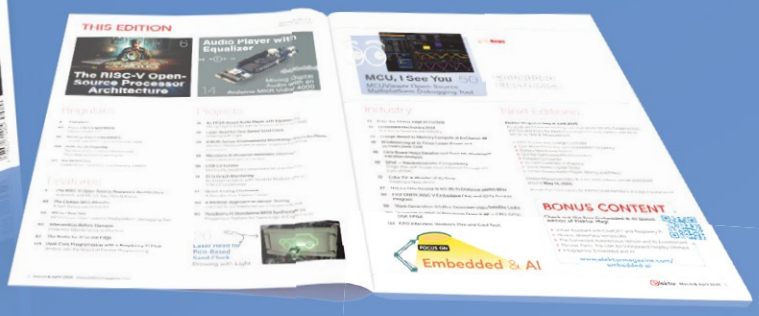


Infografik:

Prüf- und Messtechnik

Treten Sie jetzt der Elektor Community bei!

Jetzt elektor GOLD membership Mitglied werden!



- Zugang zum kompletten Online-Archiv (1970-heute)!
- 8x Elektor Magazin (gedruckt)
- 8x Elektor Magazin (PDF)
- 10% Rabatt im Elektor Store und exklusive Angebote
- Zugriff auf über 5.000 Gerber-Dateien

  u.v.m. aus der Projektplattform Elektor Labs

Auch erhältlich

Die digitale elektor GREEN membership Mitgliedschaft!

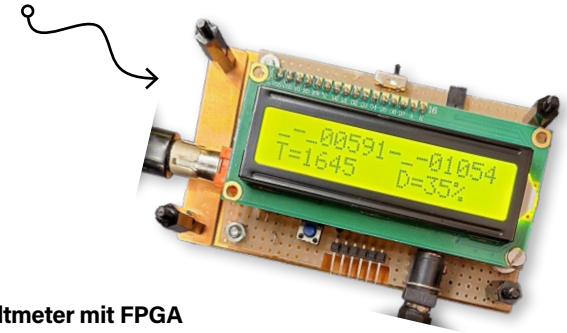
- Zugang zum kompletten Online-Archiv
- 8x Elektor Magazin (PDF)
- 10% Rabatt im Elektor Store und exklusive Angebote
- Zugriff auf über 5.000 Gerber-Dateien

  u.v.m. aus der Projektplattform Elektor Labs

www.elektormagazine.de/abo

elektor MEDIA & LEARNING

INHALT



EDITORIAL

C. J. Abate
Content Director, Elektor

Prüfen und Messen auf Ihrem Labortisch

Lösungen zum Prüfen und Messen sind entscheidend für die Fehlersuche, Validierung und Optimierung elektronischer Schaltungen. In dieser Bonus-Ausgabe stellen wir praktische und leicht zugängliche Werkzeuge vor, die Ingenieure, Studenten und Maker direkt auf dem Labortisch einsetzen können.

Diese Ausgabe bietet einen intelligenten Ansatz zur Signalanalyse mit „PWM-Messung mit einem PIC“. Lesen Sie den Artikel, um zu erfahren, wie man einen PIC16F628 zur Messung von PWM-Pulsdauern verwendet – und dabei High/Low-Zeiten und Daten über die gesamte Periode mit Präzision ermittelt.

Wir tauchen auch ein in die faszinierende Welt der FPGA-Programmierung. In „Voltmeter mit FPGA“ wird ein Spannungsmesser mit Hilfe der VHDPlus-Umgebung auf dem FPGA-Board MAX1000 realisiert. Es ist ein tolles Projekt für alle, die an einer Verbindung von Theorie und praktischem Entwurf interessiert sind.

Möchten Sie das Volumen Ihres Messgeräteparks verkleinern? Werfen Sie einen Blick auf das 2C53T, ein kompaktes 3-in-1-Instrument, das als Oszilloskop, Multimeter und Funktionsgenerator fungiert.

In dieser Bonus-Ausgabe geht es um die Verbesserung Ihres elektronischen Arbeitsablaufs. Ob Sie nun Signale feinabstimmen, Schaltungen debuggen oder einfach nur neue Testtools erforschen, wir haben alles für Sie.

Viel Spaß beim Lesen, und teilen Sie Ihre Entwicklungen auf der Labs-Plattform von Elektor!

Unser Team

Chefredakteur: Jens Nickel (v.i.S.d.P.) | **Redaktion:** Hans Adams, Asma Adhimi, Roberto Armani, Eric Bogers, Jan Buiting, Rolf Gerstendorf (RG), Ton Giesberts, Saad Imtiaz, Alina Neacsu, Dr. Thomas Scherer, Jean-Francois Simon, Clemens Valens, Brian Tristam Williams | **Regelmäßige Autoren:** David Ashton, Stuart Cording, Tam Hanna, Ilse Joostens, Prof. Dr. Martin Ossmann, Alfred Rosenkränzer | **Grafik & Layout:** Harmen Heida, Sylvia Sopamena, Patrick Wielders | **Herausgeber:** Erik Jansen | **Technische Fragen:** redaktion@elektor.de

IMPRESSUM

55. Jahrgang, Nr. 609B, ISSN 0932-5468
Mai/Juni 2025 Digitale Bonus-Ausgabe

Das Elektor Magazin wird 8 Mal im Jahr herausgegeben von
Elektor Verlag GmbH
Lukasstraße 1, 52070 Aachen (Deutschland)
Tel. +49 (0)241 95509190
www.elektor.de | www.elektormagazine.de

Chefredakteur: Jens Nickel (v.i.S.d.P.)

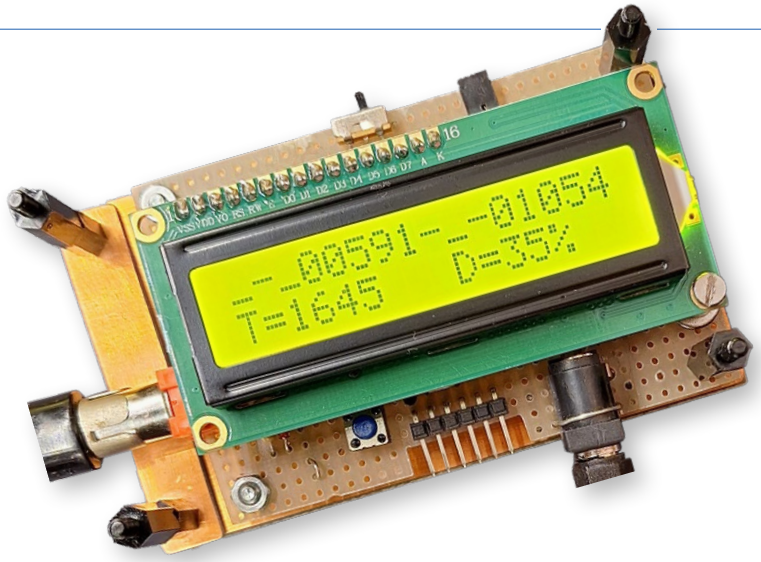
Für alle Ihre Fragen: service@elektor.de

Mitglied werden: www.elektormagazine.de/abo

Anzeigen: Büsra Kas
Tel. +49 (0)241 95509178 – busra.kas@elektor.com
www.elektormagazine.de/mediadaten

Distribution
IPS Pressevertrieb GmbH, Carl-Zeiss-Straße 5
53340 Meckenheim (Deutschland)
Tel. +49 (0)2225 88010

PWM-Messung mit einem PIC



Timing eines Rechtecksignals genau ermitteln

Von Giovanni Carrera (Italien)

Obwohl die Programmierung eines Arduino einfacher ist als die eines PICs, erweist sich letzterer aufgrund seiner Architektur als präziser bei der Berechnung des Timings eines PWM-Signals. Die zu bewältigenden Herausforderungen sind sicherlich schwieriger, aber letztendlich werden sie die Erfahrung derjenigen bereichern, die gerne programmieren. Das vorgeschlagene Gerät kann gleichzeitig hohe und niedrige Pulsdauern und auch die Periode messen, alles mit der Auflösung von einer Mikrosekunde für Pulsdauern von 10 µs bis 65.535 µs für PWM-Signale mit Frequenzen von 7,63 Hz bis 50 kHz.

Die Technik der zeitlichen Variation der Pulsdauer eines Rechtecksignals wird Pulsweitenmodulation (Pulse-Width Modulation, PWM) genannt. Diese Modulationsart, die ursprünglich aus der Telekommunikation stammt, findet heutzutage auch in der digital-analogen Welt Anwendung, um eine gemittelte Spannung oder den Strom zu variieren, deren Wert vom Verhältnis der Dauer des positiven Impulses zur Dauer der gesamten Periode abhängt. Dieser Zusammenhang wird als Tastverhältnis oder englisch Duty Cycle bezeichnet.

PWM wird zur Spannungsregelung in Schaltnetzteilen, zur Motorsteuerung, für Servoaktuatoren und sogar für das Ausgangssignal einiger Sensoren verwendet. Aus diesen Gründen verfügen die meisten modernen Mikrocontroller über einen oder mehrere PWM-Ausgänge. Einige Wandler, zum Beispiel akustische Abstandssensoren, besitzen einen digitalen Impulsausgang, dessen Dauer proportional zur Amplitude des Signals ist. Der Zweck dieses Projekts ist also nicht die Erzeugung von PWM-Signalen, sondern die präzise Messung ihrer Impulsdauer. In diesem Artikel beschreibe ich eine Schaltungsversion mit dem PIC16F628, um die Dauer des High- und des Low-Anteils des Rechtecksignals und damit auch die Periodendauer zu messen, wie im Titelbild zu sehen. Wenn Sie auch das Tastverhältnis in Prozent berechnen und anzeigen wollen, benötigen Sie einen PIC16F648, da dies eine zusätzliche Division erfordert. Das Programm benötigt dann zwar nur ein paar Befehle mehr, aber der Code überschreitet die 2K-Marge, und der F628 hat nur einen 2K großen Flash-Speicher. Quellprogramme und kompilierte HEX-Dateien stehen aber für beide Versionen unter [1] zur Verfügung. Der Rest der Hardware bleibt exakt gleich.

Der PIC-Mikrocontroller

Warum fiel die Wahl auf einen PIC-Mikrocontroller? Nun, ich hatte ein solches Gerät schon zu einer Zeit entwickelt, als ich hauptsächlich PICs in meinen Projekten verwendete, und dann später zur Entwicklung mit der Arduino-IDE und MicroPython wechselte.

Bei PIC-Mikrocontrollern ist der interne Takt ein Viertel der Frequenz des Quarzoszillators, und alle Anweisungen werden – mit Ausnahme von Programmverzweigungen – in einem Takt ausgeführt. Mit einem 16-MHz-Quarz beträgt der interne Takt 4 MHz, so dass die Ausführung eines Befehls nur 0,25 µs dauert.

Der PIC16F628 von Microchip ist ein Mikrocontroller mit so genannter RISC-Architektur (Reduced Instruction Set Computer), der mit 8-Bit-Daten arbeitet, dessen 35 Befehle aber jeweils ein 14-Bit-Wort belegen. In unserem Fall passen 2K Worte in den Flash-Speicher. Dieser PIC verfügt neben seinen zahlreichen I/O-Funktionen auch über Event Capture am Pin CCP1 (RB3), indem er den Inhalt von Timer1 in den beiden Registern CCPR1L und CCPR1H speichert. Etwas Ähnliches gibt es auch beim ATmega328P des Arduino UNO, aber die Wahl eines PICs führt zu einem viel einfacheren, kompakteren System mit viel geringerem Stromverbrauch.

Natürlich werden diejenigen, die es gewohnt sind, mit Arduino zu arbeiten, größere Schwierigkeiten haben, etwas mit PICs zu entwickeln. Es ist nicht so einfach, vorgefertigte Boards oder die vielen für Arduino

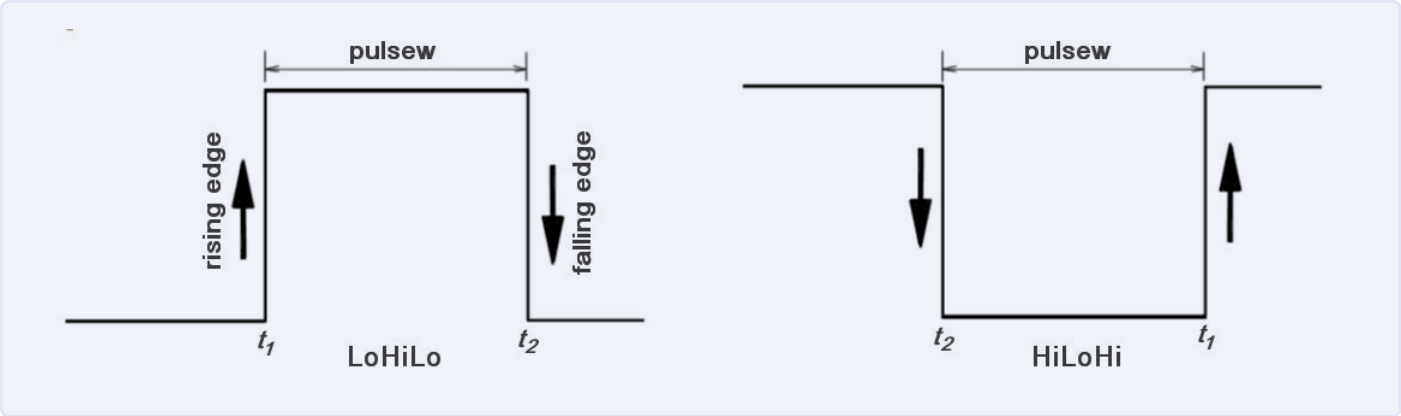


Bild 1. Es stehen zwei verschiedene Erfassungsmodi für die Impulstypen „LoHiLo" (links) und „HiLoHi" (rechts) zur Verfügung.

verfügbaren Bibliotheken zu finden, um die verwendete Peripherie zu verwalten. Meiner Meinung nach kann man bei der Arbeit mit Arduino nicht besonders tief in die Materie eindringen und weiß nicht einmal genau, was man überhaupt tut.

Viele Arduino-Benutzer machen nämlich nichts anderes, als die Verbindungen der Projekte, die aus in Fritzing- und Breadboard-Bildern angeordneten Komponenten bestehen, zu replizieren, und wissen dabei gar nicht, wie das alles funktioniert (wenn es wegen unsicherer oder falscher Verbindungen oft gar nicht funktioniert).

Der für PICs verwendete Compiler ist weniger benutzerfreundlich als die Arduino-IDE und die verfügbaren Bibliotheken sind viel spärlicher: Zum Beispiel gibt es keine `pulsein()`-Funktion, um die Dauer eines Impulses zu messen. Aus diesen Gründen muss sich der Entwickler eingehender mit der Funktionsweise von Mikrocontrollern befassen, wie ich es auch bei diesem Projekt getan habe, wo ich das Datenblatt [2] studieren musste, um die Funktionsweise von Timern und CCP-Capture zu verstehen.

Der mikroPascal-Compiler, den ich für die Programmentwicklung auf PICs verwende, ist nur für Programme kostenlos, deren Code kleiner als 2K Wörter (1 Wort = 14 Bits) ist. Diese Grenze konnte in diesem Fall auch gar nicht überschritten wurde, da der Flash-Speicher des verwendeten PICs ohnehin nicht größer ist. Der Arbeitsspeicher, das RAM, ist bei diesem PIC-Modell nur 224 Bytes groß. Mit diesen Grenzen kann man nicht viel anfangen, denn wenn man zum Beispiel eine Float-Division durchführt, wird die Speichergrenze leicht überschritten. Aber die Kompilierung verläuft viel schneller als bei Arduino, und natürlich ist der erzeugte Code auch kompakter.

Das Programm wurde in einem leistungsfähigen Pascal-Cross-Compilers für Windows-PCs namens mikroPascal PRO für PIC, Version 7.6.0 entwickelt. Eine Vollversion kann von der MIKROE-Website heruntergeladen werden [3].

In dieser ersten Version des Programms ist der Code kürzer, so dass Sie ihn ohne Probleme kompilieren können. Wer mit C oder Basic besser vertraut ist, kann entsprechende Compiler herunterladen und das Programm in die neue Sprache übersetzen. Wenn man mit dem vorgegebenen Programm zufrieden ist und keine Änderungen vornehmen möchte, spart sich die Arbeit in einem Compiler ganz und benötigt nur einen PIC-Programmierer, der die bereits kompilierte Hex-Datei *PWMmeter.hex* auf den Controller lädt.

Eine weitere Version des Programms zur Berechnung der Einschaltdauer wurde ebenfalls entwickelt, benötigt aber wegen der Fließkommadivision mehr Flash-Speicher und einen PIC16F648. In diesem Fall müssen Sie, wenn Sie die Compiler-Lizenz nicht erworben haben, den Chip mit der HEX-Datei dieses Projekts programmieren. Diese beiden Mikrocontroller sind Pin-zu-Pin-kompatibel.

Messung der Impulsdauer

Einige Mikrocontroller mit eingebautem BASIC-Interpreter wie die BASIC-Stamp, Picaxe oder BasicX verwenden eine spezielle Routine (`PULSIN`) zur Messung der Pulsdauer, können aber kurze Pulse nicht messen, weil Interpreter zu langsam sind. Arduino-Boards verwenden die Funktion `pulsein(pin, level, timeout)`, um die Dauer eines Pulses (High- oder Low-Level) mit einer Auflösung in Mikrosekunden zu messen. Für längere Zeiten gibt es die Funktion `pulseInLong()`, die Interrupts verwendet und Zeiten von 10 µs bis 3 min misst. Beide Funktionen geben einen `unsigned long` zurück, einen Long-Wert ohne Vorzeichen.

Das Programm verwendet zwei Interrupts: Timer0 für die Abtastzeit, während die zweite Interrupt-Quelle bei Capture/Compare/PWM oder kurz CCP-Erfassungsereignissen an der steigenden und fallenden Flanke des Pulses generiert wird und die entsprechenden Zeiten in Timer1 gezählt werden.

Zeitsteuerung mit Timer0

Timer0 erzeugt einen Interrupt, mit dem alle 50 Ereignisse, das bedeutet etwa 0,5 s Messwerte abgetastet und auf dem LCD angezeigt werden. Timer0 ist ein 8-Bit-TMR0-Zähler, dem ein programmierbarer 8-Bit-Teiler, der sogenannte Vorteiler, vorangestellt ist.

Das Programm wählt den internen Taktgeber mit einer Frequenz, die einem Viertel der Quarzfrequenz entspricht: 16 MHz / 4 = 4 MHz. Wenn wir den Vorteiler auf den höchstmöglichen Wert (1:256) einstellen, hat Timer0 als Eingang eine Frequenz von f1 = 4 MHz / 256 = 15,625 kHz mit einer Periode von 64 µs; das Laden von 100 in den Zähler führt zu einem Interrupt bei allen 256-100 = 156 Impulsen der Frequenz f1, was einer Periode von 156 × 64 = 9984 µs (etwa 100 Hz) entspricht.

Der Timer0-Interrupt wird erzeugt, wenn der Timer/Zähler im TMR0-Register von 0xFF auf 0x00 überläuft, wodurch das T0IF-Bit gesetzt wird. Dieser Interrupt erzeugt den Takt, mit dem die Messwerte abgetastet und auf der LCD-Anzeige dargestellt werden, und zwar alle 50 Ereignisse, was etwa 0,5 s entspricht.

Ereigniserfassung mit Timer1

Das Programm wechselt nach der steigenden Flanke am Pin RB3 in den Erfassungsmodus für die fallende Flanke und umgekehrt. Ein Impuls kann Low-High-Low-Pegel aufweisen, was wir als LoHiLo bezeichnen wollen, oder invertiert sein (HiLoHi), wie in **Bild 1** gezeigt. Bei Impulsen vom Typ LoHiLo setzt das Programm das Register `CCP1CON := $05` auf eine Erfassung bei steigender Flanke. Wenn dieses Ereignis eintritt, speichert die Interrupt-Routine den Inhalt des Registerpaars `CCPR1L` und `CCPR1H`, der dann an t1 übertragen wird. Zur Erfassung auf die fallende Flanke wird `CCP1CON := $04` gesetzt, bei

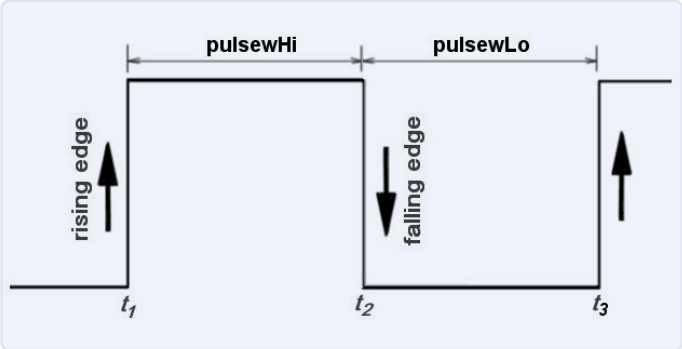


Bild 2. Erfassung der Zeitpunkte t1, t2 und t3 zur Durchführung der erforderlichen Berechnungen.

deren Eintreten das Registerpaar `CCPR1L` und `CCPR1H` erneut gelesen wird, was dann nach t2 übertragen wird, wie in Bild 1 zu sehen ist. Die `pulsew`-Zeit ergibt sich aus der Differenz t2-t1.
Bei Impulsen des Typs HiLoHi verhält es sich ähnlich, aber man beginnt mit der fallenden Flanke (t2) und endet mit der steigenden Flanke (t1). In diesem Fall ist die `pulsew`-Zeit = t1-t2. Timer1 hat einen 16-Bit-Zähler, dem ein 2-Bit-Prescaler vorangestellt ist.
Auch hier wird der interne Takt von 4 MHz verwendet. Mit dem Vorteiler = 1:1 ergibt sich eine Auflösung von 0,25 µs, was bei einem normalen Quarz zu knapp ist, so dass ein Verhältnis 1:4 mit einer Auflösung von 1 µs verwendet wurde. Daher misst das System Impulse mit einer Dauer von etwa 10 µs bis 65.535 µs. Unter 10 µs funktioniert es nicht, weil es mehrere Anweisungen in der Interrupt-Routine gibt und dies eine gewisse Ausführungszeit erfordert. Wenn Sie längere Zeiten mit einer Auflösung von 2 µs messen wollen, müssen Sie das Verhältnis 1:8 verwenden. In diesem Fall sollten die Zeiten mit zwei multipliziert werden, nachdem die Variablen in lange Ganzzahlen umgewandelt wurden.
Eine maßgeschneiderte neue und größere Version des Originalprogramms kann die beiden Dauern gleichzeitig messen; daher können Sie die PWM-Periode und auch das Tastverhältnis berechnen. In diesem Fall wählt das Programm die Erfassung für die steigende Flanke, speichert die Zeit in t1, hebt das `firstcre`-Flag auf high, um diese Flanke von der letzten zu unterscheiden, setzt dann für die fallende Flanke und speichert die Zeit in t2, und setzt für die nächste steigende Flanke, die es in t3 speichert, wie in **Bild 2** zu sehen. Bei der dritten Flanke setzt die Interrupt-Routine das `datok`-Flag, um anzuzeigen, dass die Zeiten t1, t2 und t3 gemessen wurden. Und berechnet:

pulsewHi = t2-t1, pulsewLo = t3-t2, period = pulsewHi + pulsewLo

Das Tastverhältnis erfordert Operationen mit `float`-Variablen (`real` in Pascal), die die Grenzen des Flash-Speichers des 16F628 überschreiten, aber wir können sie schnell mit einem Taschenrechner durchführen:

Periode: $T = pulsewHi + pulsewLo$
Tastverhältnis: $D = pulsewHi / T \times 100$ [%]

Vergleich mit Arduino pulseIn()

Zum Vergleich, nur für einen LoHiLo-Puls, habe ich auch ein kleines Programm für den Arduino Uno geschrieben:

```
// program pulseintest
#define pulsein 4

void setup() {
  Serial.begin(115200);
  pinMode(pulsein, INPUT);
}

void loop() {
  unsigned long duration = pulseIn(pulsein, HIGH);
  Serial.print("Pulse High [us] = ");
  Serial.println(duration);
  delay(500);
}
```

Um die Zeiten zu überprüfen, habe ich mein eigenes quarzbasiertes Gerät mit programmierbarem Teiler und Rechteckausgang als Referenz verwendet. **Tabelle 1** vergleicht die Ergebnisse mit denen meines PIC-Systems. Wie man sehen kann, ist das PIC-System sehr genau.

Table 1: Messungen

ref. [µs]	PIC [µs]	Arduino [µs]	Fehler % PIC	Fehler % Arduino
5		5...6		0...20
50	50	50...51	0	0...2
500	500	494...500	0	−0,8...0
5.000	5.000	4.966	0	−0,68
50.000	49.997	49.662	−0,006	−0,676

Das Schaltbild

Bild 3 zeigt den Schaltplan des Systems. Bei einem Spannungsregler LM7805 (ein 78L05 tut es auch) liegt der Eingangsspannungsbereich zwischen 7 V (optimaler Wert) und 12 V. Das Gerät benötigt aber (ohne Hintergrundbeleuchtung) weniger als 10 mA, sodass die Verlustleistung des Reglers vernachlässigt werden kann. Viel bequemer macht man es sich mit einer Versorgung durch drei gewöhnliche 1,5-V-Batterien. Kritisch ist in diesem Fall vielleicht das LCD mit einer Nennversorgungsspannung von 5 V, aber solche Displays funktionieren normalerweise auch mit 4,5 V.

Stückliste

Widerstände:
R1 = 10 k, ±5%, ¼ W
R2 = 100 Ω, ±5%, ¼ W
R3 = 100 Ω, ±5%, ½ W
Rp1 = 22 k, Trimmpoti

Kondensatoren:
(alle keramisch)
C1, C2 = 22 p, 50 V
C3, C4, C5, C6 = 100 n, 25 V

Halbleiter:
U1 = PIC16F628A oder PIC16F648A (Mikrocontroller)
U2 = LM7805 (5-V-Regler)
D1, D2 = 1N4148 (Diode)

Außerdem:
Anzeige = LCD 2×16
X1 = 16-MHz-Quarz
SW1 = Drucktaster (RESET)

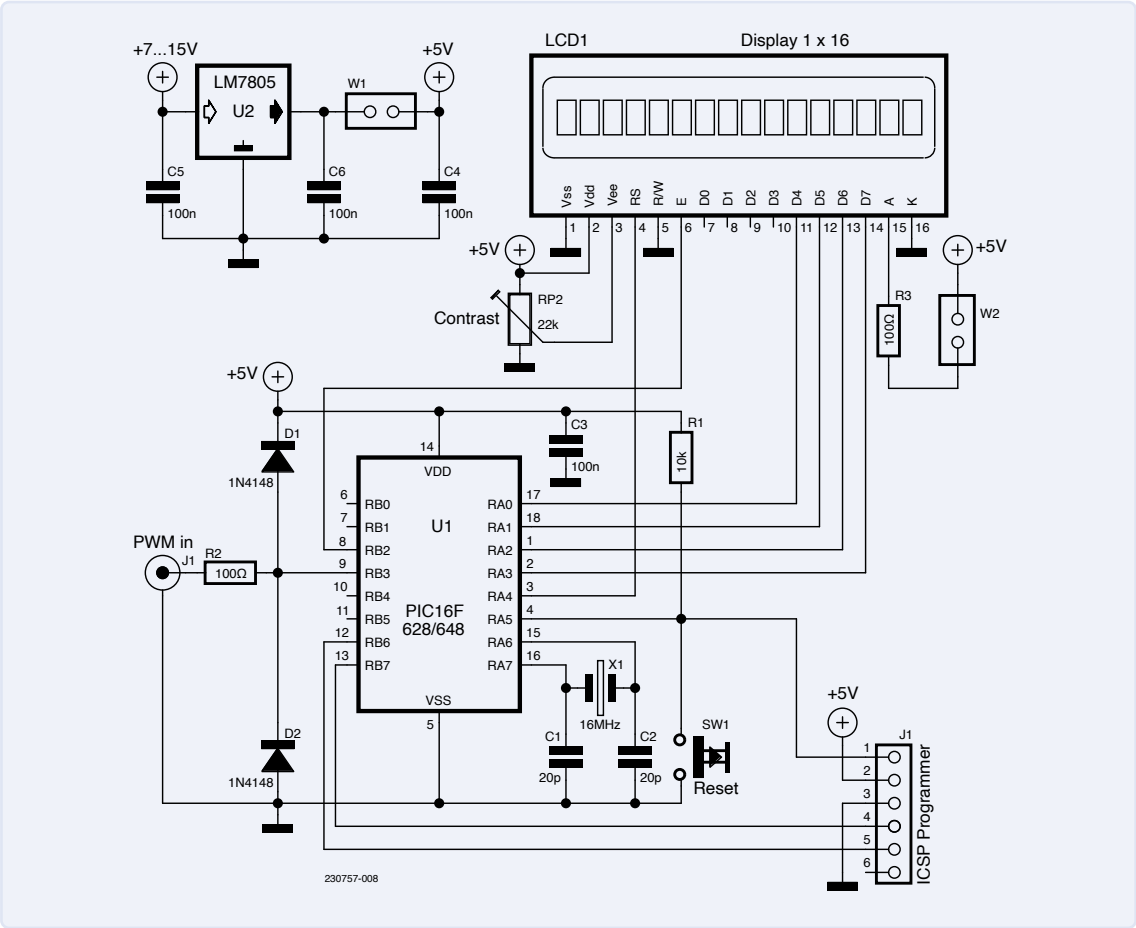


Bild 3. Schaltplan des Projekts.

Das Display habe ich ein handelsübliches zweizeiliges 16-Zeichen-LCD mit einer HD44780-kompatiblen Parallelschnittstelle verwendet. Der Jumper W1 verhindert Konflikte mit der Stromversorgung und muss entfernt werden, wenn Sie den Chip mit dem PICKIT programmieren. Jumper (oder Schalter) W2 wird verwendet, um die Hintergrundbeleuchtung abzuschalten und Strom zu sparen.

Der Prototyp

Bild 4 zeigt die Anordnung der Bauteile auf der Lochrasterplatine des Prototyps. Die LC-Anzeige wurde entfernt, um die darunter angebrachten Bestandteile der Schaltung zu zeigen.

Oben ist der kleine Schalter W2 zu sehen, der die Hintergrundbeleuchtung des Displays ausschaltet, während links ein Cinch-Stecker für den PWM-Eingang angebracht ist und unten ein 6-poliger Steckverbinder für die Chip-Programmierung, der mit dem PICkit-Programmiergerät kompatibel ist. Wenn Sie ein PIC-Programmiergerät mit einer ZIF-Fassung haben, ist dieser Anschluss nicht erforderlich. Sie müssen dann aber den PIC aus seiner Fassung ziehen, um ihn zu programmieren, was oft mit verbogenen Beinchen verbunden ist, wenn man nicht aufpasst. Neben dem Verbinder, immer noch an der Unterseite, ist der Stromanschluss zu sehen. Die Platine wurde aus einem älteren Projekt recycelt, zu dem auch die Buchsenleiste rechts gehörte und die hier keine Funktion hat.

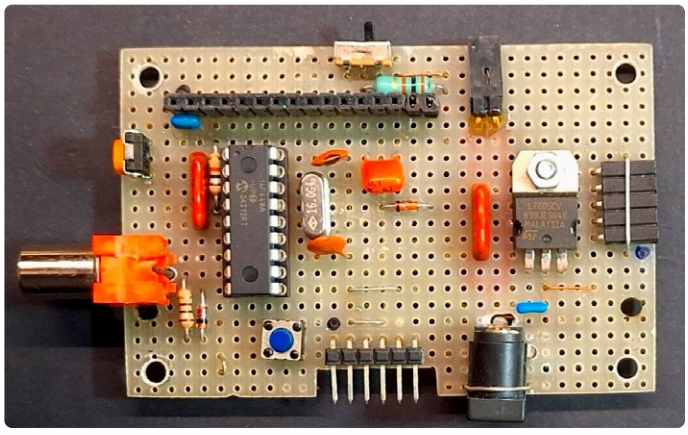


Bild 4. Der fertige Prototyp, aufgebaut auf einem Breadboard aus einem früheren Projekt. Die 5-polige Buchsenleiste auf der rechten Seite ist nicht in Gebrauch.

Das Programm

Der größte Teil des Programms wurde bereits oben beschrieben. Die Ausgaben auf dem LCD sind grundlegender als die Funktionen der Arduino-Bibliothek *LiquidCrystal*, und die Meldungen müssen Strings oder Arrays von Zeichen sein, so dass es eine notwendige Aufgabe des Programms sein muss, die Variablen in Strings umzuwandeln und alle führenden Leerzeichen zu entfernen.

Die wichtigste Arbeit wird von der Interrupt-Routine erledigt. Im ersten Teil prüft sie, ob die Unterbrechung durch einen Timer0-Überlauf verursacht wird. In diesem Fall setzt `INTCON`, `T0IF` = 1 das `int0flg`-Flag high, das für das Timing verwendet wird. Anschließend wird die Zahl 100 in den Zähler geladen, so dass er nach 156 Impulsen einen Interrupt auslöst und die Interrupt-Flags `INTCON` und `T0IF` auf 0 zurücksetzt.

Der zweite, komplexere Teil ist die Erfassung der steigenden und fallenden Flanken und die Speicherung der Inhalte der beiden Erfassungsregister `CCPR1L` und `CCPR1H` in den Variablen `t1`, `t2` und `t3`, die der ersten steigenden Flanke, der fallenden Flanke und der zweiten steigenden Flanke entsprechen. Wenn alle drei Zeiten erfasst wurden, wird das `datok`-Flag gesetzt, das vom Programm zur Berechnung der Dauern und der Periode verwendet wird.

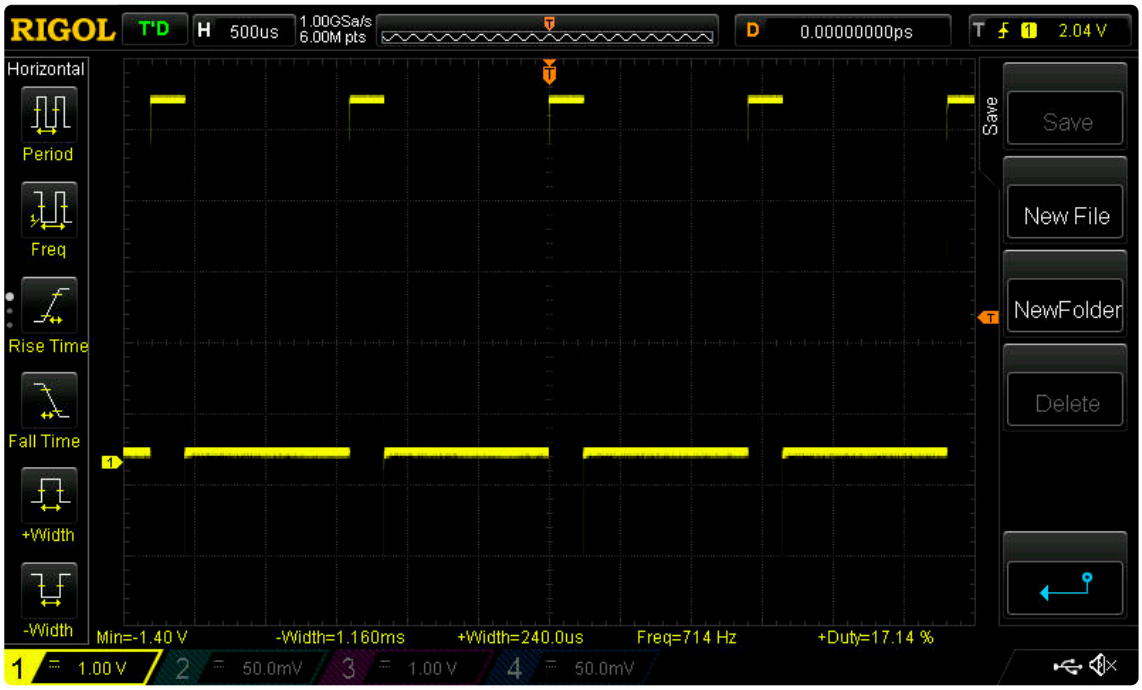


Bild 5. Oszilloskop-Screenshot der Messung aus Bild 6.

Die in Mikrosekunden ausgedrückten Zeiten für den High-Anteil der PWM, gekennzeichnet durch `_-_`, und für den Low-Anteil, der mit `-_-` gekennzeichnet ist, werden in der ersten Displayzeile ausgegeben. Die PWM-Periode, also die Summe der beiden Zeiten, ist in der zweiten Zeile angegeben.
Wenn kein Signal anliegt, zeigt das Display nichts an. Der vollständige Code ist in **Listing 1** zu sehen. Dieses Programm belegt 1.236 Wörter des Flash-Speichers (65 %) und 74 Byte des RAM (37 %), und die Kompilierung wurde in 187 ms durchgeführt.

Programm mit Tastverhältnis-Anzeige

In diesem Fall benötigen Sie einen PIC16F648A, der dem F628 ähnlich ist, aber 4K Worte Flash-Speicher besitzt. Wie bereits erwähnt, müssen Sie zum Kompilieren dieser Version die nicht ganz billige Programmlizenz erwerben, die die 2K-Wort-Begrenzung aufhebt. Die bereits kompilierte Hex-Datei für den PIC16F648 ist jedoch im Paket für diesen Artikel enthalten und kann direkt in den PIC geladen werden.
Das Programm gibt `pulsewHi` [µs], `pulsewLo` [µs], `period` [µs] und `dutyc` [%] auf dem Display aus.
Leider hat mir die Fließkommadivision bei der Kompilierung Speicherprobleme bereitet, da auch der PIC16F648 dafür nicht besonders geeignet ist. Ich habe die Unannehmlichkeiten überwunden, indem ich die prozentuale Einschaltdauer in eine ganze Zahl umgewandelt habe, wobei natürlich der Dezimalteil verloren ging.
Das Programm selbst ist dem vorherigen weitgehend ähnlich, mit Ausnahme der `LCDprint`-Routine. Es wurden Anweisungen zur Berechnung und zur Ausgabe des Tastverhältnisses hinzugefügt. Die geänderte Routine ist in Listing 2 dargestellt.
Der geänderte Code belegt 2.266 Wörter des Flash-Speichers (55 %) und 92 Byte des RAM (62 %), und die Kompilierung wurde in 47 ms durchgeführt.

Zum Schluss ein Vergleich

Bild 5 und **Bild 6** zeigen einen Vergleich zwischen Messungen mit einem Oszilloskop und denen des PIC16F648-Systems. In diesem Fall wurde ein TTL-Pulsgenerator verwendet. Wie Sie sehen können, sind die Messungen sehr ähnlich.

RG — 230757-02

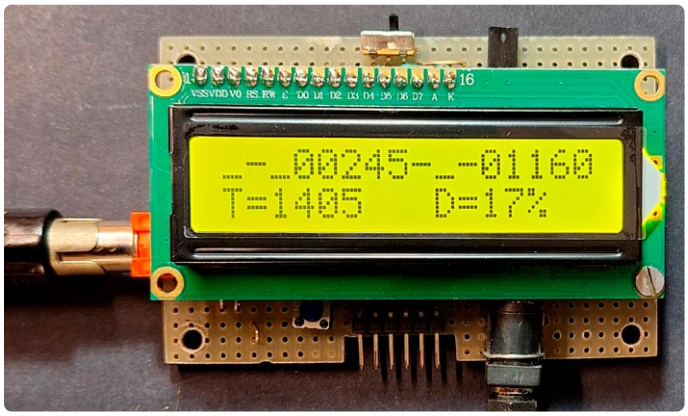


Bild 6. Anzeige der High-Pulsdauer (245 µs), des Low-Pulses (1.160 µs), der Gesamtperiode T (1.405 µs) und des Tastverhältnisses (17 %).

Listing 2: PIC16F648 PWMmeter (Auszug)

```
procedure LCDprint; // print measures on LCD
begin
  Lcd_Cmd(_LCD_CLEAR); // lcd clear
  WordToStrWithZeros(pulsewHi, texdH);// microseconds
  ltrim(texdH); // trims the leading spaces
  WordToStrWithZeros(pulsewLo, texdL);
  ltrim(texdL); // trims the leading spaces
  Lcd_Out(1, 1,'_-_' + texdH + '-_-' + texdL);
  LongWordToStr(period, texPer);
  ltrim(texPer); // trims the leading spaces
  //FloatToStr_FixLen(dutyc, texd, 5);
  Lcd_Out(2, 1,'T=' + texPer);
  dutyc:= real(pulsewHi)/real(period)*100.0;
  IntToStr(integer(dutyc), texd);
  ltrim(texd);
  Lcd_Out(2, 10,'D=' + texd + '%');
  LCDout:= false;
end;
```

Über den Autor

Giovanni Carrera hat einen Abschluss in Elektrotechnik. Als Universitätsprofessor an der Fakultät für Schiffstechnik in Genua, Italien, unterrichtete er in zahlreichen Kursen zum Beispiel über Schiffsautomatisierung und die Simulation von Schiffsantriebssystemen. Carrera begann in den späten 1970er Jahren mit der 6502-CPU zu arbeiten und ging dann zu anderen Prozessoren über. Heute widmet er sich dem Entwurf und der Entwicklung analoger und digitaler elektronischer Schaltungen, über die er in seinen Blogs (ArduPicLab und GnssRtkLab) und in verschiedenen Zeitschriften geschrieben hat.

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Bitte kontaktieren Sie die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **Tam Hanna, *Microcontroller Basics with PIC*, Elektor 2020**
  E-Buch, PDF, englisch: www.elektor.de/19188
- **Microchip MPLAB PICkit 5 In-Circuit Debugger/ Programmierer**
  www.elektor.de/20665

WEBLINKS

[1] Software-Download für dieses Projekt: https://elektormagazine.de/230757-02
[2] Datenblatt PIC16F627A/628A/648A: https://ww1.microchip.com/downloads/en/DeviceDoc/40044G.pdf
[3] MIKROE-Website: https://mikroe.com/mikropascal-pic

Listing 1: PIC16F628 PWMmeter

```
program PWMmeter;
// PIC16F628 measures PWM (resolution= 1 usec)
// display the value on LCD every 0.5 sec
// timer#0 for interrupt every 9.984 msec (approx 100Hz)
// the LCD display has  2 rows x 16
// by G. Carrera 23/12/2023
// I/O configuration:
// PortB.3 = pulse input   (in)
// PortA.0..3,= LCD data (4 bit mode) (out)
// PortB.2 = LCD E (out)
// PortA.4 = LCD RS (out)
// xtal = 16.00 MHz
var
  t1,t2,t3: word;
  int0flg,datok,LCDout,firstcre: boolean;
  i,t1l,t1h,t2l,t2h,t3l,t3h: byte;
  pulsewLo,pulsewHi: word;
  texdL: string[5];
  texdH: string[5];
  period: longint;
  texPer: string[10];
  dutyc: real;
  texd: string[6];
```

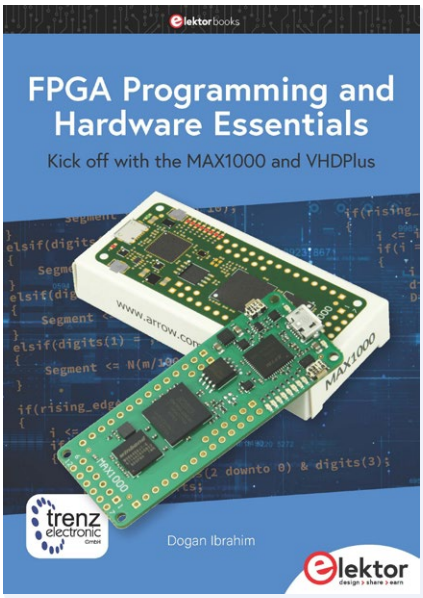
(Fortsetzung auf der nächsten Seite)

```
// Lcd module connections
var LCD_RS : sbit at RA4_bit;
var LCD_EN : sbit at RB2_bit;
var LCD_D4 : sbit at RA0_bit;
var LCD_D5 : sbit at RA1_bit;
var LCD_D6 : sbit at RA2_bit;
var LCD_D7 : sbit at RA3_bit;
var LCD_RS_Direction : sbit at TRISA4_bit;
var LCD_EN_Direction : sbit at TRISB2_bit;
var LCD_D4_Direction : sbit at TRISA0_bit;
var LCD_D5_Direction : sbit at TRISA1_bit;
var LCD_D6_Direction : sbit at TRISA2_bit;
var LCD_D7_Direction : sbit at TRISA3_bit;
// End Lcd module connections
procedure interrupt;
// read capture times between rising and falling edge or vice versa
begin
  if TestBit(INTCON, T0IF) then  // timer#0 interrupt
    begin
      ClearBit(INTCON, T0IF); // reset Timer#0 interrupt flag
      TMR0:= 100;
      SetBit(INTCON, T0IE); // Enables Timer#0 interrupt
      int0flg:= true;
    end;
  if TestBit(PIR1, CCP1IF) then // capture interrupt
    begin
      if  TestBit(CCP1CON, CCP1M0)  then // rising edge ($05)
        begin
          if firstcre then  // already captured the first rising edge t1
            begin
              t3l := CCPR1L;  //  load final time to t3
              t3h := CCPR1H;
              firstcre:= false;
              datok:= true;
            end
          else
            begin
              t1l := CCPR1L;  //  load initial time to t1
              t1h := CCPR1H;
              firstcre:= true; // the first rising edge is captured now
            end;
          CCP1CON := $04; // change to falling edge on CCP1
          ClearBit(PIR1, CCP1IF); //Clear CCP1 Int. flag (also due to changing)
        end
      else  // falling edge
        begin
          t2l := CCPR1L;  //  load intermediate time to t2
          t2h := CCPR1H;
          CCP1CON := $05; // change to rising edge on CCP1
          ClearBit(PIR1, CCP1IF); //Clear CCP1 Int. flag (also due to changing)
        end;
    end;
end;
procedure IOinit; // initialize I/O
begin
  TRISB:= %00111011;  // PORTB bit2 : output
  TRISA:= %10100000;  // PORTA bits 0..4 are outputs
  CMCON:= $07; // comparators off (RA0..RA4 usable as digital IO)
  Lcd_Init(); // Initialize LCD
  Lcd_Cmd(_LCD_CURSOR_OFF); // Turn off cursor
```

(Fortsetzung auf der nächsten Seite)

```
  CCP1CON := $05;  //  rising edge on CCP1
  T1CON := $21;   // 1:4 prescaler, int. clk, enable t1
  OPTION_REG:= $07; // set prescaler of Timer#0 to 256
  ClearBit(OPTION_REG,7); //  enable port B pull-ups
  SetBit(PIE1,CCP1IE); // Enables the CCP1 interrupt
  SetBit(INTCON, PEIE); // Enables peripheral interrupt
  TMR0:= 100; // T0 overflow every 156 counts
  SetBit(INTCON, T0IE); // Enables Timer#0 interrupt
  ClearBit(INTCON, T0IF); // reset Timer#0 interrupt flag
  SetBit(INTCON, GIE); // Enables global interrupt
  int0flg:= false;
  Lcd_Out(1, 1, 'PWMmeter-231223'); // Print text to LCD (row,column,text)
  Delay_ms(2000);
  Lcd_Cmd(_LCD_CLEAR); // lcd clear
  i:=0;
  firstcre:= false;
end;
procedure LCDprint; // print measures on LCD
begin
  Lcd_Cmd(_LCD_CLEAR); // lcd clear
  WordToStrWithZeros(pulsewHi, texdH);// microseconds
  ltrim(texdH); // trims the leading spaces
  WordToStrWithZeros(pulsewLo, texdL);
  ltrim(texdL); // trims the leading spaces
  Lcd_Out(1, 1,'_-_' + texdH + '-_-' + texdL);
  LongWordToStr(period, texPer);
  ltrim(texPer); // trims the leading spaces
  //FloatToStr_FixLen(dutyc, texd, 5);
  Lcd_Out(2, 1,'T=' + texPer);
  LCDout:= false;
end;
begin  // main program
  IOinit; // Initialize
  LCDout:= false;
  while true Do //  endless loop
    begin
      if int0flg then
        begin
          int0flg:= false;
          i:=i+1;
          if i= 50 then // about 0.5 seconds
            begin
              LCDout:= true;
              i:= 0;
            end;
        end;
      if datok then
        begin
          datok:= false;
          t1:= (t1h shl 8)+t1l;
          t2:= (t2h shl 8)+t2l;
          t3:= (t3h shl 8)+t3l;
          pulsewHi:= t2-t1;
          pulsewLo:= t3-t2;
          period:= longint(pulsewHi) + longint(pulsewLo);
          if LCDout then LCDprint;
        end;
    end;
end
```

Voltmeter mit FPGA



MAX1000 und VHDPlus machen es einfach

Von Dogan Ibrahim (Großbritannien)

Sind Sie bereit für die FPGA-Programmierung? Das Elektor-Buch *FPGA Programming and Hardware Essentials* bietet einen idealen Einstieg in die spannende Welt der feldprogrammierbaren Gatterarrays. Als kurzes Beispiel betrachten wir ein Voltmeter, das mit der leistungsstarken VHDPlus-Programmierumgebung entworfen und auf dem MAX1000-FPGA realisiert wurde.

Anmerkung der Redaktion: *Dieser Artikel ist ein Auszug aus dem Elektor-Buch „*FPGA Programming and Hardware Essentials*". Er wurde so formatiert und leicht bearbeitet, damit er den Konventionen und dem Seitenlayout der Zeitschrift Elektor genügt. Für Rückfragen stehen der Autor und die Redaktion gerne zur Verfügung. Die Kontaktdaten finden Sie im Textkasten* **Fragen oder Kommentare?**

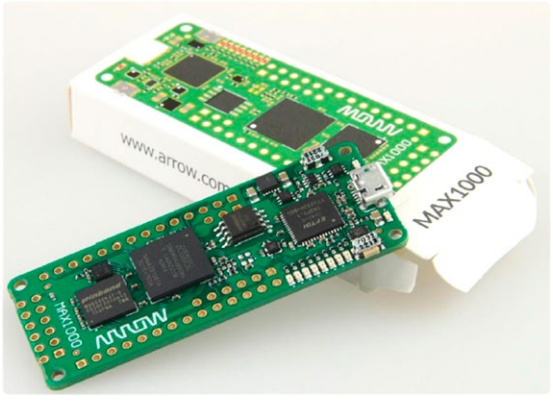


Bild 1. Das FPGA MAX1000.

Ein FPGA ist eine konfigurierbare integrierte Schaltung, die programmiert werden kann, um verschiedene Aufgaben zu übernehmen, die normalerweise einzelnen Chips zugeordnet sind. Wenn Sie dachten, das Programmieren eines FPGAs zu Hause sei schwierig und nur Profis möglich, dann … lesen Sie weiter!

Lernen Sie das FPGA MAX1000 kennen!

Das im Buch hauptsächlich genutzte FPGA ist das MAX1000-Entwicklungsboard von Arrow Electronics (**Bild 1**). Es wurde entwickelt, um den Einstieg in die Nutzung eines FPGAs zu erleichtern. Obwohl das Board in anderen Kapiteln des Buches behandelt wird, sollen hier zur Einführung seine grundlegenden Eigenschaften kurz genannt werden.

- Intel MAX10
- Arrow USB-Programmer 2
- 64 Mbit SDRAM
- 64 Mbit Flash-Speicher
- 1× 12-MHz-MEMS-Oszillator
- 1× optionaler MEMS-Oszillator der bevorzugten Frequenz

- 8× Nutzer-LEDs (rot)
- 2× Board-Anzeige-LEDs
- 2× Nutzer-Tasten
- 1× 3-Achsen-Beschleunigungssensor
- 1× 12-poliger Pmod-Steckverbindung (nur Footprint)
- 1× Nutzer-JTAG-Verbinder (nur Footprint)
- 1× Nutzer-I/O-Steckverbindung (nur Footprint)
- Mini-USB-Anschluss Typ-B

Für die Software: die VHDPlus-IDE

Es stehen verschiedene Softwarepakete zur Verfügung, um FPGA-Anwendungen auf dem Entwicklungsboard für das MAX1000-FPGA zu entwickeln. In diesem Buch lernen Sie die grundlegenden Funktionen sowie die Installation der VHDPlus-IDE kennen, mit dem Sie Ihren MAX1000 programmieren.
Die VHDPlus-IDE stellt eine erweiterte Version von VHDL dar, die das Programmieren erleichtert, indem sie die Funktionen erweitert und die Syntax vereinfacht. VHDPlus ist keine völlig neue Sprache, sondern basiert auf den Möglichkeiten von VHDL. Alles, was mit VHDL möglich ist, kann also auch mit VHDPlus umgesetzt werden.
VHDPlus bietet einen modernen Ansatz, der die FPGA-Programmierung schneller und benutzerfreundlicher macht, besonders für Einsteiger. Die VHDPlus-IDE unterstützt den offenen CRUVI-Standard. Die Simulation eines FPGA-Programms kann zeitaufwendig sein, doch der Simulationsassistent in der VHDPlus-IDE hilft Ihnen, Programme zu simulieren und Fehler schnell zu beheben. Die VHDPlus-IDE besitzt wichtige Funktionen aus Quartus und ist sowohl für Windows als auch für Linux verfügbar. Außerdem unterstützt die VHDPlus-IDE C++ und enthält einen Debugger.

VHDPlus-IDE installieren

Die Schritte zum Herunterladen der VHDPlus-IDE auf einem Windows-Computer sind wie folgt:

- Besuchen Sie die VHDPlus-Website [1].
- Klicken Sie auf **Schritt 1**, um die Datei *MAX 10 device support* herunterzuladen **(Bild 2**).
  Die Datei wird in Ihrem *Downloads* -Ordner gespeichert. Legen Sie sie in einem eigenen Ordner ab.
- Gehen Sie zu **Schritt 2** und laden Sie *Quartus Prime Lite for Windows* herunter. Auch diese Datei wird in Ihrem *Downloads* -Ordner gespeichert.
- Klicken Sie auf die Datei, um das Programm *Quartus Lite* zu installieren.
- Beim Start von *Quartus Lite* könnte folgende Fehlermeldung erscheinen: *You successfully installed the Quartus Prime software but did not install any devices. Do you want to launch the device installer to add devices?* Installieren Sie die Gerätesupportdatei .qdz wie unten beschrieben.
- Gehen Sie zum Windows-*Start*-Menü und suchen Sie nach dem *Geräteinstallationsprogramm* (Quartus Prime 18.1).
- Das *Geräteinstallationsprogramm* wird Sie nach dem Ordner fragen, in dem die .qdz-Datei gespeichert wurde. Wählen Sie diesen Ordner aus.

Updates vom Autor
Seit der Veröffentlichung des Buches und als Reaktion auf das Feedback der Leser haben wir vom Autor zwei Updates erhalten, die sich hauptsächlich mit dem verwendeten FPGA-Typ befassen.

1. Bitte beachten Sie Seite 34, zweiter Punkt: Ein Fenster öffnet sich (**Bild 3.6**), in dem Sie das Entwicklungsboard auswählen können, das Sie verwenden.

- Dieses Fenster trägt den Titel *Connect and Compile*
- Oben links sehen Sie den Text: *FPGA:* gefolgt von einer Dropdown-Liste
- Wählen Sie *MAX1000* aus der Liste für die 8-kLE-Variante (Standardoption)
- Wählen Sie *MAX1000 16k* aus der Liste für die 16-kLE-Variante

Wichtig: Die Auswahl der falschen Variante führt zu einem JTAG-Fehler, wenn Sie versuchen, den kompilierten Code auf das FPGA zu laden.

2. Die 8-kLE-Version ist durch die Bezeichnung *10M08* auf dem großen Chip auf dem Board gekennzeichnet, während die 16-kLE-Version mit *10M16* markiert ist.

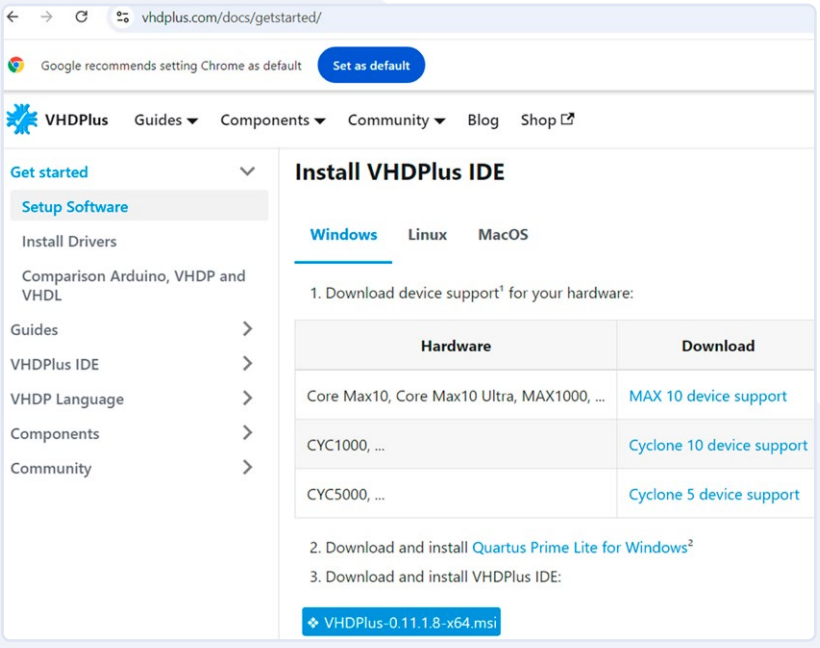


Bild 2. Laden Sie die Datei MAX 10 device support herunter.

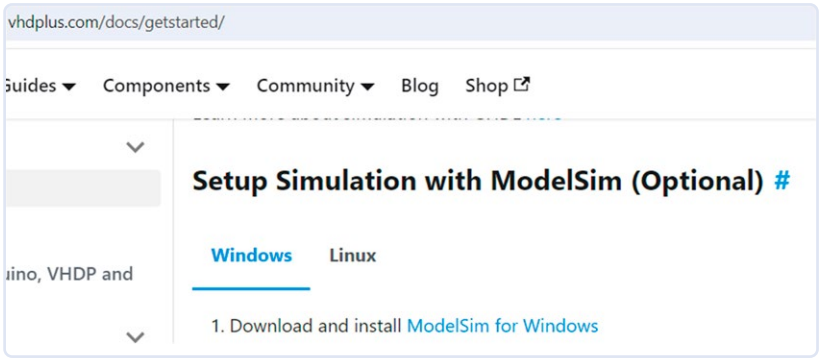


Bild 3. Laden Sie den Simulator ModelSim herunter.

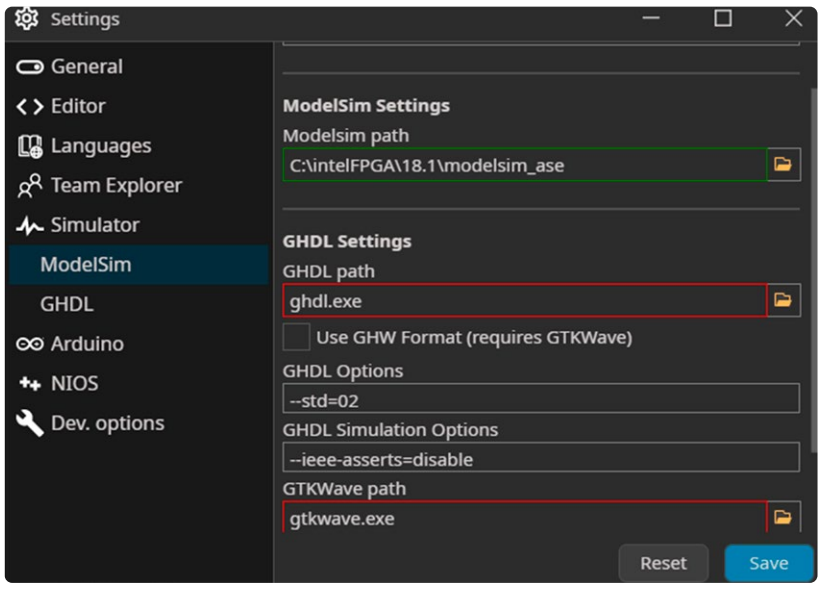
- Das Programm sucht nach allen .qdz-Dateien. Wählen Sie das Gerät aus, das Sie installieren möchten. Die .qdz-Datei ist zunächst nur ein Installationspaket, bis Sie sie installieren.
- Klicken Sie auf **Schritt 3**, um VHDPlus herunterzuladen. Die Datei wird in Ihrem *Downloads*-Ordner gespeichert.
- Klicken Sie auf die Datei, um VHDPlus zu installieren.

Sie müssen die Treiber für Ihren Programmierer installieren, um Ihr FPGA programmieren zu können. Folgen Sie diesen Schritten:

- Laden Sie den Treiber *Arrow USB Programmer* für Ihr Betriebssystem von [2] herunter.
- Entpacken Sie die heruntergeladene Datei und führen Sie das Installationsprogramm aus, um die Installation abzuschließen.

Für das direkte Kompilieren und Programmieren mit der VHDPlus-IDE ist eine Verbindung zu Quartus erforderlich. Wenn Quartus im Standardverzeichnis installiert ist, sind keine weiteren Schritte nötig.

Bild 4. Geben Sie den Pfad zum Simulator in der VHDPlus-IDE an.



Andernfalls passen Sie den Quartus-Pfad in der VHDPlus-IDE an, indem Sie zu *Extras, Settings* und schließlich *General* gehen. Sobald Quartus erkannt wird, wird der Rahmen grün.

Einrichten des Simulators ModelSim

Um den Simulator ModelSim zu installieren, folgen Sie bitte diesen Schritten:

- Besuchen Sie die VHDPlus-Website [1].
- Scrollen Sie nach unten und klicken Sie auf den Download-Link für *ModelSim for Windows* (**Bild 3**). Die Datei wird in Ihrem Downloads-Ordner gespeichert.
- Installieren Sie *ModelSim*, indem Sie die heruntergeladene Datei doppelklicken.
- Geben Sie in der VHDPlus-IDE den Pfad zum *modelsim_ase* -Ordner unter *Extras* → *Settings* → *Simulator* an (**Bild 4**).
- Führen Sie eine Simulation durch, indem Sie mit der rechten Maustaste auf eine VHDL-Datei klicken.

Analog-Digital-Wandler

Analog-Digital-Wandler (ADCs) sind entscheidende Bestandteile vieler Anwendungen, die auf Mikrocontrollern basieren. Dies liegt daran, dass die meisten physikalischen Sensoren analoge Ausgangsspannungen erzeugen. Ein analoger Temperatursensor kann beispielsweise eine Spannung ausgeben, die direkt proportional zur gemessenen Temperatur ist. Lassen Sie uns daher die analogen Eingänge des FPGAs MAX1000 betrachten und ein Projekt mit diesen Eingängen entwickeln.

Das FPGA MAX1000 hat neun analoge Eingänge, die AIN sowie A0…A7 heißen. Jeder Eingang hat eine Auflösung von 12 Bit, was 4096 Stufen entspricht. Mit der Standard-Referenzspannung von +3,3 V beträgt die Auflösung eines ADC-Eingangs also 3,3 V / 4096 = 0,8 mV. Somit können analoge Eingangs-

Tabelle 1: ADC-Kanäle des MAX1000

Analog-Pin	Kanal
AIN0	1
AIN1	6
AIN2	5
AIN3	7
AIN4	3
AIN5	0
AIN6	2
AIN7	4
AIN	8

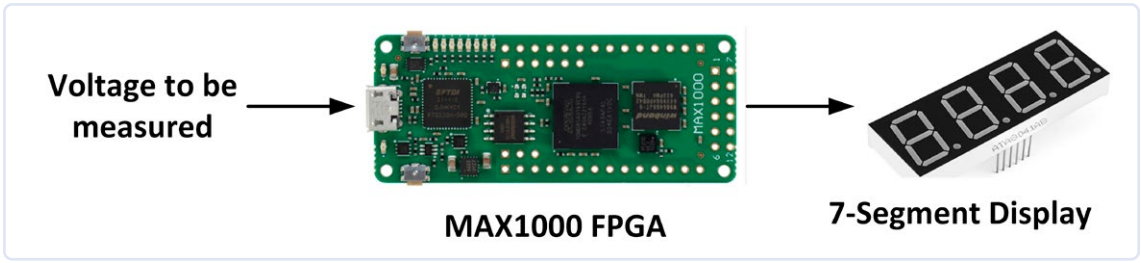


Bild 5. Blockschaltbild des FPGA-Voltmeters.

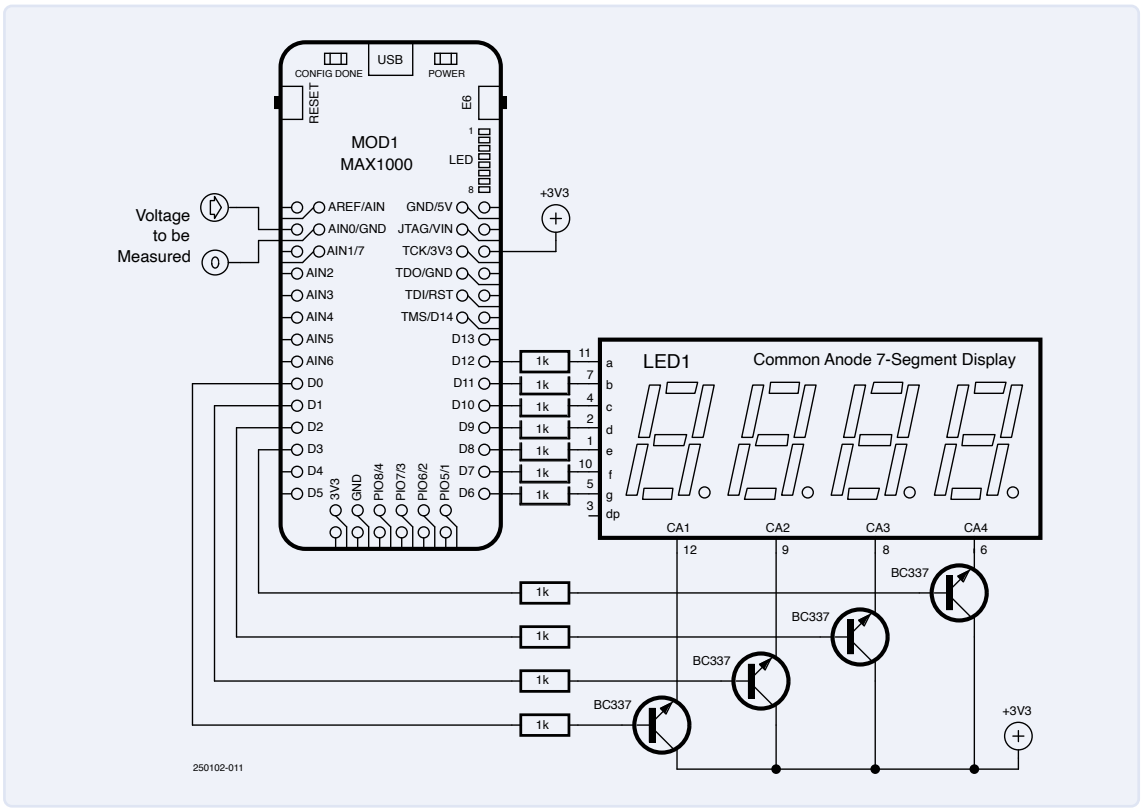


Bild 6. Schaltplan des FPGA-basierten Voltmeters.

spannungen mit einer Genauigkeit von 0,8 mV erfasst werden. Wenn die Eingangsspannung zum Beispiel 0,8 mV beträgt, zeigt der ADC-Ausgang binär *0000 00000001* an. Liegt die Eingangsspannung bei 0,16 mV, gibt der ADC-Ausgang 0000 00000010 aus. Bei einer Eingangsspannung von 2,4 mV zeigt der ADC-Ausgang *0000 00000011* und so weiter.
Die analogen Eingänge sind als Kanäle vorhanden. In **Tabelle 1** sind die Kanalnummern der einzelnen analogen Eingänge aufgeführt.

Projekt Voltmeter

Hardware

Lassen Sie uns als erstes Projekt eine einfache Voltmeter-Schaltung erstellen, die die an einem der ADC-Eingänge angelegte Spannung misst und diese auf einem 7-Segment-Display in Millivolt anzeigt. Die maximale Eingangsspannung beträgt +3,3 V (3300 mV). **Bild 5** zeigt das Blockdiagramm des Projekts; die Schaltung

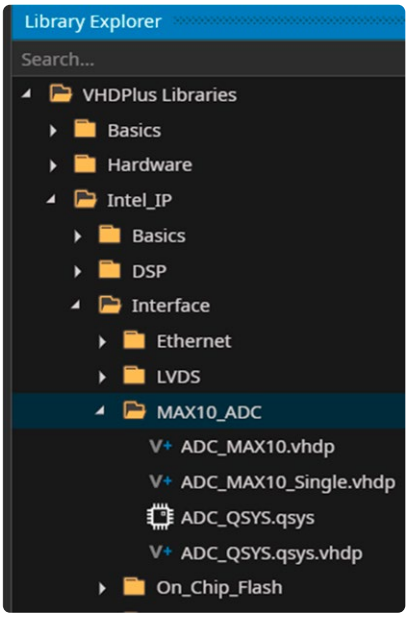


Bild 7. Fügen Sie die ADC-Bibliothek zu Ihrem Projekt hinzu.

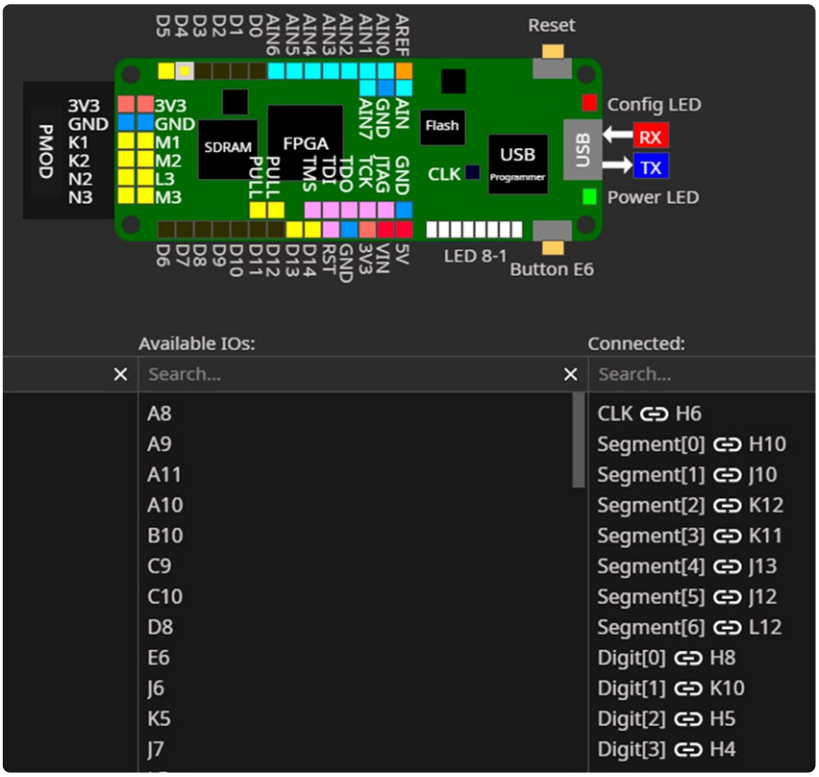


Bild 9. FPGA-Verbindungsdiagramm.

◀ *Bild 8. Bestätigung, dass die ADC-Bibliothek zum Projekt hinzugefügt wurde.*

ist in **Bild 6** dargestellt, wobei die zu messende Spannung am ADC-Eingang AIN0 (Kanal 1) anliegt.

Software

Wir müssen die *ADC*-Bibliothek verwenden, um die ADC-Eingänge zu nutzen. Die Schritte sind wie folgt:

- Öffnen Sie die IDE und benennen Sie Ihr Programm (zum Beispiel *Voltmeter*).
- Klicken Sie in der unteren linken Ecke des Fensters auf *Library Explorer*.
- Wählen Sie *Intel_IP Interface MAX10_ADC* (**Bild 7**):
- Klicken Sie mit der rechten Maustaste auf *MAX10ADC* und wählen Sie *Add to active project*. Die *ADC*-Bibliothek wurde nun zu unserem Projekt hinzugefügt. Unten im Fenster sollte eine Nachricht erscheinen, dass die *ADCMAX10 ADC*-Bibliothek zum aktiven Projekt hinzugefügt wurde **(Bild 8)**.

Jetzt können Sie beginnen, Ihren Code mit der Komponente *ADC_MAX10_Single* aus der *ADC*-Bibliothek zu schreiben.

Bild 9 zeigt das Diagramm mit der Verbindung des 7-Segment-Displays. Eine ausführlichere Behandlung erfolgt in einem anderen Kapitel des Buches.

Das Programmlisting *Voltmeter* ist im Kasten **Listung 1** dargestellt. Der Abschnitt mit dem 7-Segment-LED-Display ist identisch mit dem, den wir in Kapitel 5 behandelt haben. Zusätzlich haben wir die Signale `temp`, `ADCdata`, und `ADCchannels` definiert.

Listing 1: Der VHDPlus-Code sorgt dafür, dass das Voltmeter auf einem FPGA läuft

```
Main
(
   Segment: OUT STD_LOGIC_VECTOR(6 downto 0);
   Digit: OUT STD_LOGIC_VECTOR(3 downto 0);
)

{
     TYPE MyArray is an array (0 to 9) of STD_LOGIC_VECTOR(1 to 7);
     SIGNAL N: MyArray;
     SIGNAL m: integer range 0 to 9999 := 0;
     SIGNAL temp: integer range 0 to 3300 := 0;
```

```
    SIGNAL digits: STD_LOGIC_VECTOR(3 downto 0) := "0001";
    SIGNAL i: integer range 0 to 36000 := 0;
    SIGNAL ADCdata: natural range 0 to 4095 := 0;
    SIGNAL ADCchannels: natural range 0 to 8 := 0;

    NewComponent ADC_MAX10_Single
    (
        Channel => ADCchannels,
        Data => ADCdata
    );

    N(0) <= "1000000";
    N(1) <= "1111001";
    N(2) <= "0100100";
    N(3) <= "0110000";
    N(4) <= "0011001";
    N(5) <= "0010010";
    N(6) <= "0000010";
    N(7) <= "1111000";
    N(8) <= "0000000";
    N(9) <= "0010000";

  Process()
  {
     if(digits(0) = '1')
     {
        Segment <= N(m mod 10);
     }
     elsif(digits(3) = '1')
     {
        Segment <= N(m/10 mod 10);
     }
     elsif(digits(2) = '1')
     {
        Segment <= N(m/100 mod 10);
     }
     elsif(digits(1) = '1')
     {
        Segment <= N(m/1000 mod 10);
     }
     if(rising_edge(CLK))
     {
      i <= i + 1;
      if(i = 36000)
      {
           i <= 0;
           digits <= digits(2 downto 0) & digits(3);
           Digit <= digits;
      }
      ADCchannels <= 1;            -- Read channel 1 (AIN0)
      temp <= ADCdata;             -- As raw data
      m <= temp * 3300 / 4095;     -- in millivolts
     }
  }
}
```

Das Signal `temp` hat einen Wertebereich von 0...3300, was der maximalen Spannung am Ausgang des ADC entspricht. `ADCdata` sind die rohen ADC-Daten, die Werte von 0...4095 annehmen können, wobei 4095 dem Binärwert *1111 11111111* entspricht. `ADCchannels` ist die Kanalnummer und kann Werte von 0...8 annehmen.

Ein neues Element namens `ADC_MAX10_Single` wird im Programm definiert, um auf das einzelne ADC-Modul in der ADC-Bibliothek zuzugreifen. Dieses Element enthält zwei Variablen in der Bibliothek: Kanalnummer (`Channel`) und Kanal-Data (`Data`).

Bei der ansteigenden Taktflanke wird Kanal 1 ausgewählt, der dem ADC-Eingang AIN0 entspricht. Die Rohdaten des ADC werden in das Signal `temp` kopiert. Dieser Wert wird mit 3300 multipliziert und durch 4095 geteilt, um die Rohmessung in physikalische Millivolts umzuwandeln. Wenn dieser Wert nach `m` kopiert wird, zeigt er den ADC-Wert an, also die Voltmetermessung im 7-Segment-Display.

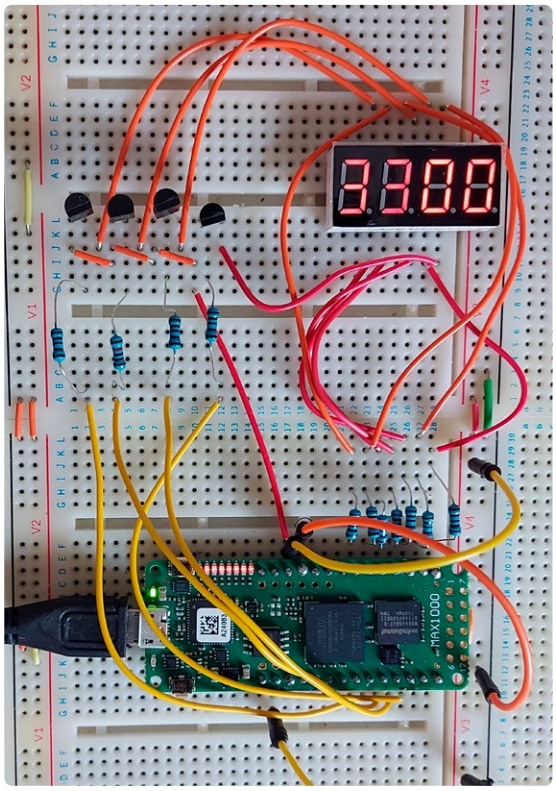


Bild 10. Das Voltmeter-Projekt, aufgebaut auf einem Breadboard.

Testen wir es!

Bild 10 zeigt das Projekt, das vom Autor für die Veröffentlichung im Buch erstellt wurde. Obwohl es in seinen Möglichkeiten begrenzt ist, stellt dieses kleine Voltmeter eine ausgezeichnete Einführung in die spannende Welt der FPGA-Programmierung dar, und zwar mit kostengünstiger sowie leicht zugänglicher Hardware und Software. Was ist Ihr nächstes Projekt?

Übersetzung: Mahy Arafa — 250102-02

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zum Artikel? Wenden Sie sich bitte an den Autor unter d.ibrahim@btinternet.com oder Elektor unter redaktion@elektor.de.

Über den Autor

Prof. Dr. Dogan Ibrahim besitzt einen Bachelor in Elektrotechnik, einen Master in Automatisierungstechnik und einen Doktortitel in digitaler Signalverarbeitung. Bevor er in die akademische Welt zurückkehrte, arbeitete er in verschiedenen Industrieunternehmen. Prof. Ibrahim hat über 70 Fachbücher geschrieben und mehr als 200 technische Artikel über Mikrocontroller, Mikroprozessoren und ähnliche Themen veröffentlicht. Er ist ein zugelassener Elektroingenieur und Fellow of the Institution of the Engineering Technology. Außerdem ist er ein zertifizierter Arduino-Experte.

Passende Produkte

- **Dogan Ibrahim, *FPGA Programming and Hardware Essentials* (Elektor 2024)**
  Taschenbuch, englisch: www.elektor.de/21054
  E-Buch, PDF, englisch: www.elektor.de/21055
- **Dogan Ibrahim, MAX1000 FPGA Programming Bundle (Elektor 2024)**
  Buch 21054 und MAX1000-Board: www.elektor.de/21082

WEBLINKS

[1] VHDPlus-Webseite: https://vhdplus.com/docs/getstarted/#install-vhdplus-ide
[2] Treiber für Arrow-USB-Programmer, Trenz Electronic: https://tinyurl.com/Arrow-USB-Programmer

WERDEN SIE MITGLIED UNSERER COMMUNITY

KOSTENLOSER DOWNLOAD

Melden Sie sich heute an, **elektormagazine.de/ezine-24**

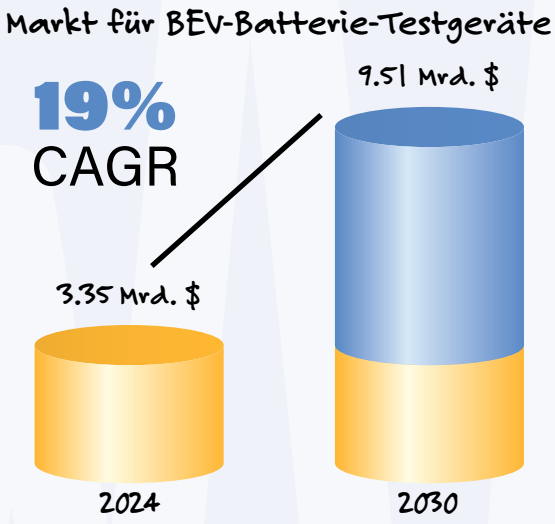
elektor MEDIA & LEARNING

BEV-Batterietests: Der wichtigste Wachstumstreiber

Auf dem Markt für Batterie-Testgeräte für batteriebetriebene Elektrofahrzeuge (BEV) steigt die Nachfrage nach spezialisierten Testeinrichtungen wie Batterietestern und Wärmemanagementsystemen zur kritischen Bewertung von BEVs [1]. Der Bereich der batterieelektrischen Fahrzeuge dominiert und treibt das Wachstum von EV-Prüfgeräten aufgrund der besonderen Prüfanforderungen an. Da im Jahr 2023 rund 40 Millionen Elektrofahrzeuge auf den Straßen unterwegs waren [2], sind für BEVs umfangreiche Tests der Batterie, des Antriebsstrangs und der Ladeschnittstelle erforderlich, um Leistung, Effizienz und Sicherheit zu gewährleisten. Fortschritte bei der Batterietechnologie, einschließlich der Festkörper- und Hochleistungs-Lithium-Ionen-Chemie, erfordern spezielle Protokolle, um die Effizienz der Batterien zu validieren.

Das Segment der Leistungstests wird bis zum Jahr 2033 voraussichtlich am schnellsten wachsen. Unternehmen wie der TÜV Süd sind in diesem Bereich führend und bieten Dienstleistungen wie zyklische und kalendarische Alterungstests, Schnellladeprofile und elektrochemische Impedanzspektroskopie an, um für eine optimale Batterieleistung zu sorgen [3].

Hauptakteure
- TÜV Süd
- Bureau Veritas
- SGS Societe Generale De Surveillance SA
- TÜV Rheinland AG
- Dekra
- Applus+

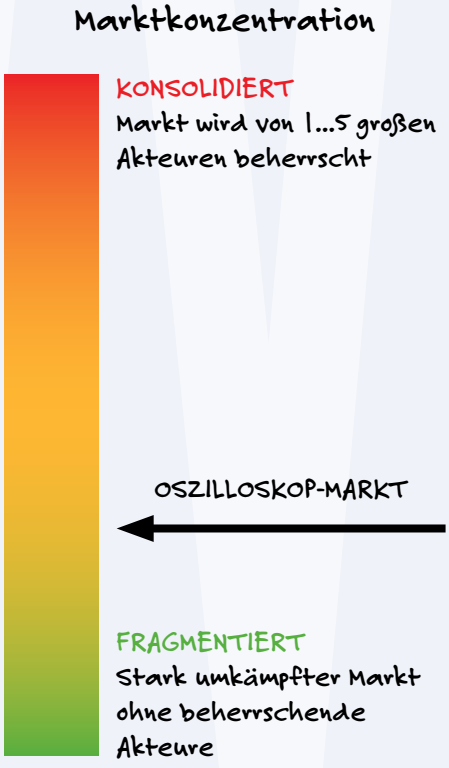


Quelle: Research and Markets [3]

Oszilloskope bleiben an der Spitze

Der Markt für allgemeine Prüf-und Testgeräte (General Purpose Test Equipment, GPTE) verzeichnet ein stetiges Wachstum, das auf die steigende Nachfrage nach Präzisionsmessgeräten in verschiedenen Branchen zurückzuführen ist. Laut Technavio [7] wird erwartet, dass der GPTE-Markt von 2023 bis 2028 um 2,06 Mrd. US-Dollar wächst, wobei allein der Markt für Oszilloskope, der den größten Anteil innerhalb des GPTE-Marktes hält, voraussichtlich von 3,74 Mrd. Dollar im Jahr 2025 auf 5,49 Mrd. Dollar im Jahr 2030 wachsen dürfte, bei einem jährlichen Wachstum von 7,99 % [8].

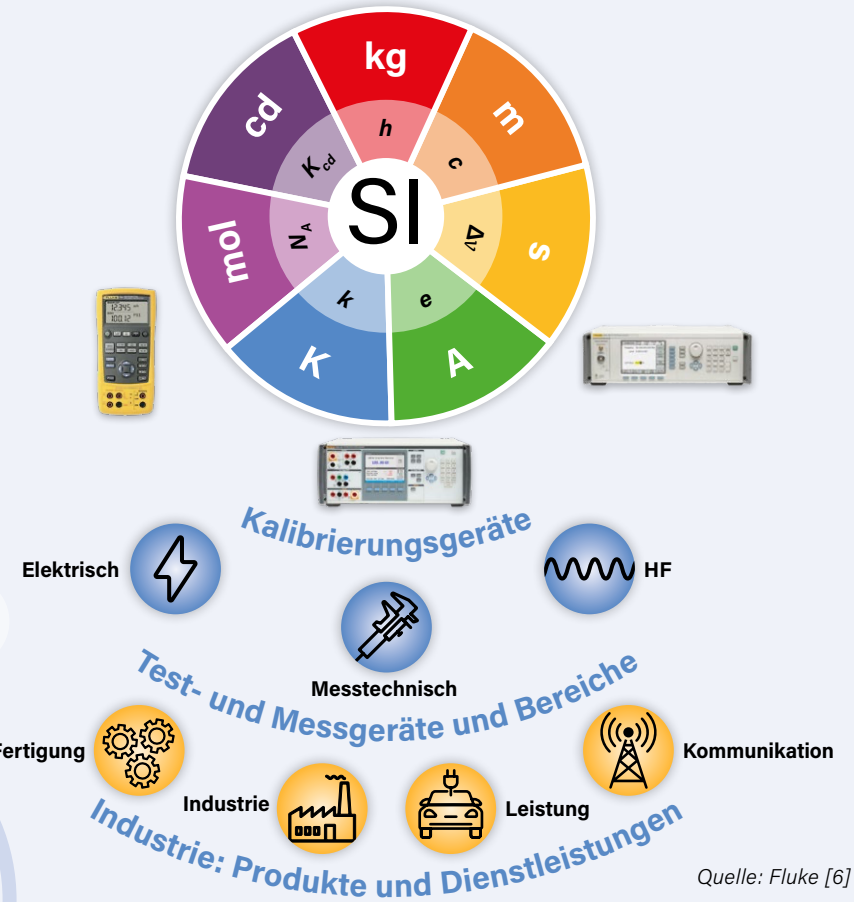
Oszilloskop-Marktführer
- Tektronix
- Keysight Technologies
- Rohde & Schwarz
- Teledyne LeCroy
- Yokogawa Test & Measurement Corporation



Quelle: Mordor Intelligence [8]

Das **Kalibrierungsuniversum**

Mit der Verschärfung der Qualitätsanforderungen in der Präzisionsfertigung wächst der Markt für Kalibrierungs- und Eichdienstleister, und es wird zwischen 2025 und 2033 eine jährliche Wachstumsrate (CAGR) von 4,5 % erwartet [4]. Die Metrologie gewährleistet die Genauigkeit sowohl in der Vor- als auch in der Nachproduktion und sorgt für die Einhaltung enger Toleranzen. Die sich weiterentwickelnden Kalibrierungsstandards erhöhen jedoch die Komplexität und stellen die Dienstleister vor die Herausforderung, immer einen Schritt voraus zu sein [5]. Gleichzeitig könnte das Aufkommen von selbstkalibrierenden Geräten und Softwarelösungen die traditionellen Dienstleistungen drastisch verändern und die Anbieter zur Anpassung zwingen.



Quelle: Fluke [6]

Übersetzung: Rolf Gerstendorf — 250108-02

WEBLINKS

[1] Towards Automotive, „Electric Vehicle Test Equipment Market Size, Trends and Developments“, September 2024: https://tinyurl.com/EV-market
[2] IEA, „Trends in electric cars“, 2024: https://www.iea.org/reports/global-ev-outlook-2024/trends-in-electric-cars
[3] Research and Markets, „EV Battery Testing Market“," Dezember 2024: https://www.researchandmarkets.com/reports/6036229
[4]] IMARC Group, „Calibration Services Market“, 2024: https://www.imarcgroup.com/calibration-services-market
[5] MarketsandMarkets, „Calibration Services Market“, 2023: https://tinyurl.com/mam-calibration-market
[6] Fluke, „Why is Calibration Important?“: https://www.fluke.com/en-us/learn/blog/calibration/why-is-calibration-important
[7] Technavio, „General Purpose Test Equipment (GPTE) Market“, August 2024: https://tinyurl.com/technavio-GPTE-market
[8] Mordor Intelligence, „Oscilloscope Market“, 2024: https://www.mordorintelligence.com/industry-reports/oscilloscope-market

Oszilloskop/ Multimeter/ Generator

Fnirsi 2C53T mit zwei Kanälen und 50-MHz-Bandbreite

Bild 1. Das 2C53T von Fnirsi ist kompakt und ersetzt drei separate Messgeräte.

Von Harry Baggen (Niederlande)

Der Strom neuer Produkte des chinesischen Herstellers Fnirsi scheint nicht abzureißen. Diesmal schauen wir uns das 2C53T an, ein kompaktes 3-in-1-Messgerät, das ein Oszilloskop, ein Multimeter und einen Funktionsgenerator vereint. Es sieht genauso aus wie das 2C23T, das ich vor etwa einem Jahr getestet habe. Was ist also neu?

Das Fnirsi 2C53T [1] ist ein kompaktes 3-in-1-Messgerät, das ein Oszilloskop, ein Multimeter und einen Funktionsgenerator kombiniert (**Bild 1**). Der chinesische Hersteller Fnirsi bringt immer wieder neue Produkte auf den Markt. Manchmal handelt es sich dabei um verbesserte Versionen bereits veröffentlichter Geräte, und das ist bei dem hier vorgestellten 2C53T der Fall. Äußerlich sieht das Gerät genauso aus wie das 2C23T, das ich vor etwa einem Jahr untersucht habe [2].

Gedächtnisauffrischung

Das Gehäuse bleibt gleich, mit nur kleinen Änderungen bei der Beschriftung einiger Tasten. Für alle, die das Vorgängermodell nicht kennen, hier sind die wichtigsten Merkmale des 2C23T und des 2C53T von Fnirsi. Das Gerät hat ein kompaktes Gehäuse mit den Maßen 17 × 9 × 3,5 cm und ist mit einem 2,8-Zoll-Farb-LCD ausgestattet, das ein helles Bild mit gutem Kontrast zeigt. Die Ecken des Gehäuses sind mit blauem, gummiartigem Material versehen und fühlen sich stabil an. Die Stromversorgung erfolgt über einen integrierten 3-Ah-Lithium-Akku, der für etwa sechs Stunden Betrieb ausreicht.

An der Vorderseite befinden sich vier Bananenbuchsen für das Multimeter und an der Oberseite des Geräts drei BNC-Buchsen: zwei für das (zweikanalige) Oszilloskop und eine für den Funktionsgenerator. Die Bedienung des Geräts erfolgt über 15 Drucktasten. Ein USB-C-Anschluss an der Seite ermöglicht das Aufladen des internen Lithium-Akkus und den Anschluss an einen PC, um die Firmware zu updaten und Screenshots zu speichern.

Viel Zubehör im Lieferumfang enthalten

Das 2C53T wird jetzt in einem praktischen Aufbewahrungskoffer geliefert, der Platz für die Multimeter-Messleitungen, zwei 100-MHz-Oszilloskop-Tastköpfe (das 2C23T hatte nur einen), ein BNC-Kabel mit Krokodilklemmen für den Generator, ein USB-C-Kabel und ein kleines Handbuch bietet (**Bild 2**). Wie bei den meisten Fnirsi-Geräten ist alles ordentlich verarbeitet und gut verpackt. Der Aufbewahrungskoffer ist stabil und durchdacht gestaltet.

Bild 2. Das 2C53T wird unter anderem mit zwei Oszilloskop-Tastköpfen in einem praktischen Aufbewahrungskoffer geliefert.

Unterschiede zwischen 2C53T und 2C23T

Mit dem 2C53T [1] wurden die Spezifikationen und Möglichkeiten der Oszilloskopfunktion erheblich verbessert und erweitert. Die Eingangsbandbreite wurde von 10 MHz auf 50 MHz erhöht und die Abtastrate wurde von 50 Msamples/s auf 250 Msamples/s gesteigert (**Bild 3**). Das ist außergewöhnlich für ein Messgerät in dieser Preisklasse! Außerdem verfügt das Oszilloskop jetzt über einen X-Y-Modus und bietet die Auswahl von acht mathematischen Funktionen. Es gibt auch eine FFT-Analyse, die allerdings für ein Gerät dieser Art nicht besonders nützlich ist.

Es wurde eine so genannte „Persistenzfunktion" hinzugefügt, bei der Signale für eine bestimmte Zeit „nachglühen", ähnlich wie bei alten Oszilloskopen mit Kathodenstrahlröhren. Die Anzahl der Messwerte, die auf dem Bildschirm angezeigt werden können, wurde auf 14 pro Kanal erweitert. Schließlich wurden alle Einstellungen des Oszilloskops in einem eigenen Menü zusammengefasst, was die Navigation in den Menüs sehr viel übersichtlicher macht.

Multimeter-Updates

An dem Multimeter hat sich nicht viel geändert. Die Auflösung wurde von vier Ziffern (9999) auf 4½ Ziffern (19999) erhöht, und die Anzeige wurde in einem schlankeren, übersichtlicheren Layout neu gestaltet. Es gibt jetzt nur noch eine Analogskala (**Bild 4**). Die technischen Daten und Funktionen sind gleich geblieben: eine Grundgenauigkeit von 0,5 %, die Möglichkeit, Gleich- und Wechselspannung und -strom sowie Widerstand, Kapazität und Temperatur zu messen. Ein Durchgangsprüfer und ein Diodentester sind ebenfalls enthalten.

Wie bei den meisten Fnirsi-Multimetern erkennt das Gerät automatisch, ob Gleichspannung, Wechselspannung oder Widerstand an den Eingängen anliegt, aber Sie können auch manuell auf eine bestimmte Funktion umschalten.

Änderungen am Funktionsgenerator

Am Funktionsgenerator wurden einige Änderungen vorgenommen. Die Anzahl der verfügbaren Wellenformen ist von sechs auf 13 gestiegen. Das Ausgangssignal hat jetzt eine maximale Aussteuerung von 3 V_{Spitze} (zuvor 3,3 V). Allerdings wurde der Frequenzbereich auf 50 kHz reduziert (im Vergleich zu 1 MHz beim 2C23T). Nun, irgendwo muss man ja Kompromisse machen.

Erste Schritte mit dem Fnirsi 2C53T

Wenn Sie schon einmal mit einem 2C23T gearbeitet haben, werden Sie sich beim 2C53T schnell zurechtfinden. Abgesehen von ein paar kleinen Änderungen bleibt die Bedienung gleich. Wenn Sie das Gerät einschalten, sehen Sie ein Menü mit verschiedenen Symbolen, über die Sie ein Messgerät auswählen oder die Einstellungen aufrufen können. Außerdem können Sie das 2C53T so einstellen, dass es direkt mit einem bestimmten Messgerät startet. Das Oszilloskop ist der wichtigste Teil dieses Geräts, vor allem wegen seiner großen Bandbreite und hohen Abtastrate. Leider wurde das Display nicht vergrößert, sodass es immer noch recht überladen wirkt, besonders wenn mehrere Messwerte angezeigt werden. Das Oszilloskop reagiert schnell auf Signaländerungen, und die Bedienung wurde durch ein neues Einstellungsmenü ein wenig verbessert. Die Einstellung der Zeitbasis, der Empfindlichkeit, des Triggerpegels und der Signalposition auf dem Bildschirm ist jetzt einfacher: Mit der Select-Taste können Sie auswählen, welcher dieser Parameter mit den Cursorkreuz-Tasten eingestellt werden kann. Die gewählten Parameter erscheinen am oberen Rand des Bildschirms.

Es ist schwer festzustellen, ob die Auto-Setup-Funktion aktualisiert wurde, aber ich hatte den Eindruck, dass sie etwas schneller arbeitet als beim Vorgängermodell. Die BNC-Anschlüsse an der Oberseite liegen immer noch sehr eng beieinander, was bedeutet, dass Tastköpfe mit Kunststoffabdeckungen nicht passen dürften.

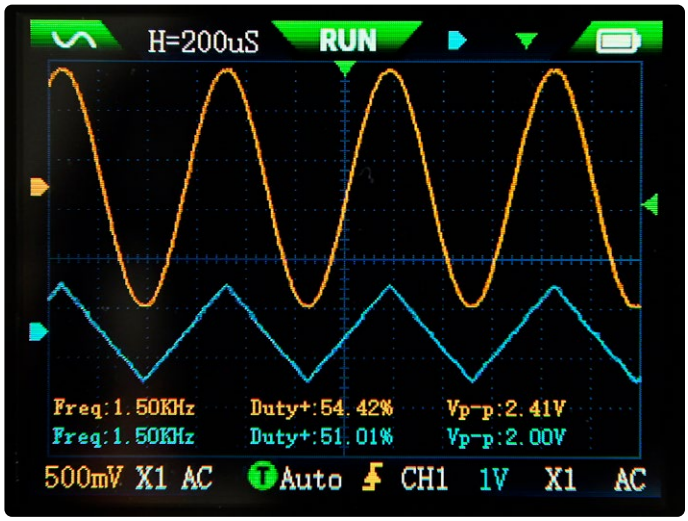


Bild 3. Das Oszilloskop ist nun bis zu 50 MHz einsetzbar und scheint auch schneller zu reagieren als das 2C23T.

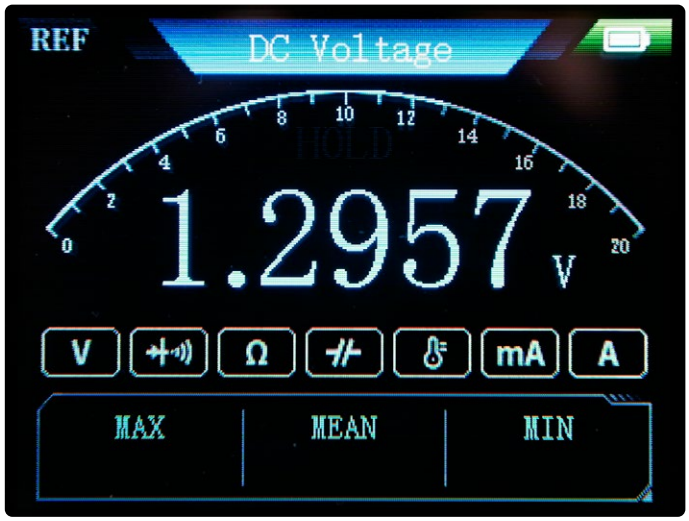


Bild 4. Das Display des Multimeters ist jetzt schlichter als zuvor gestaltet und zeigt nur noch eine analoge Skala (und das reicht aus).

Glücklicherweise sind zwei passende Tastköpfe im Lieferumfang enthalten.
Die Bandbreite des Oszilloskops beträgt mindestens 50 MHz. Es ist jedoch zu beachten, dass die Signalamplitude im Bereich 10…50 MHz leicht ansteigt, was zu kleinen Schwankungen in den Messwerten führt. Vermutlich liegt das an einer leicht nichtlinearen Eingangsstufe. Wenn Sie das wissen, können Sie es entsprechend ausgleichen.

Leistung des Multimeters

Wie das 2C23T verfügt auch das Multimeter des Fnirsi 2C53T über eine automatische Erkennung, die jedoch erst ab 0,7 V funktioniert. Bei Bedarf kann der Modus manuell umgeschaltet werden, wenn der Verdacht besteht, dass der angezeigte Wert falsch ist oder wenn ein Widerstandswert anstelle eines Spannungswerts angezeigt wird. Ich finde, dass die Anzeige des Multimeters eine Verbesserung im Vergleich zum 2C23T darstellt, da es jetzt eine einzige Analogskala mit dem Messwert enthält. Die minimalen, maximalen und durchschnittlichen Messwerte werden am unteren Rand angezeigt. Ich habe die Genauigkeit des Multimeters bei verschiedenen Spannungs-, Strom-, Widerstands- und Kapazitätswerten getestet und die Ergebnisse waren identisch mit denen des 2C23T. Tatsächlich ist die Genauigkeit mindestens doppelt so gut wie die angegebenen Spezifikationen, was sehr beeindruckend ist!
Allerdings stelle ich die Entscheidung in Frage, die Anzahl der Ziffern zu erhöhen, da dies bei einer Grundgenauigkeit von 0,5 % nicht unbedingt erforderlich ist. Für vergleichende Messungen könnte dies jedoch nützlich sein. Das Messgerät enthält zwei interne Sicherungen für die beiden Strombereiche, die nach Aufschrauben des Gehäuses ausgetauscht werden können.
Schließlich ist es etwas enttäuschend, dass die vier Eingangsbuchsen des Multimeters, wie bei seinem Vorgänger, etwas weniger als die standardmäßigen 19 mm voneinander entfernt sind, wodurch es mit einigen Standardzubehörteilen nicht kompatibel ist. Fnirsi hätte das Gehäuse wahrscheinlich etwas breiter gestalten müssen, um die richtigen Abstände zu einzuhalten.
Obwohl der Funktionsgenerator jetzt mehr Optionen bezüglich der Wellenform bietet, hat sich die Signalqualität nicht verbessert. Die Sinuskurve erscheint im unteren Bereich eindeutig etwas abgeflacht (**Bild 5**). Bei einer Vergrößerung sind die relativ groben Stufen sichtbar, aus denen das Signal besteht. Deswegen eignet es sich für allgemeine Tests, bei denen die Signalqualität keine Priorität hat. Besonders auffällig ist die Einschränkung des Frequenzbereichs auf 50 kHz, was eine deutliche Reduzierung im Vergleich zum 2C23T darstellt.
Die Funktionsweise des Generators bleibt unverändert. Die Frequenz kann weiterhin aufs Hertz genau eingestellt werden. Die maximale Ausgangsspannung beträgt jetzt 3 V im Leerlauf. Die Ausgangsimpedanz wurde angepasst und beträgt jetzt etwa 50 Ω. Beachten Sie jedoch, dass das Ausgangssignal gegenüber der Masse völlig positiv ist, da es keinen Ausgangskondensator und keine symmetrische Stromversorgung gibt.

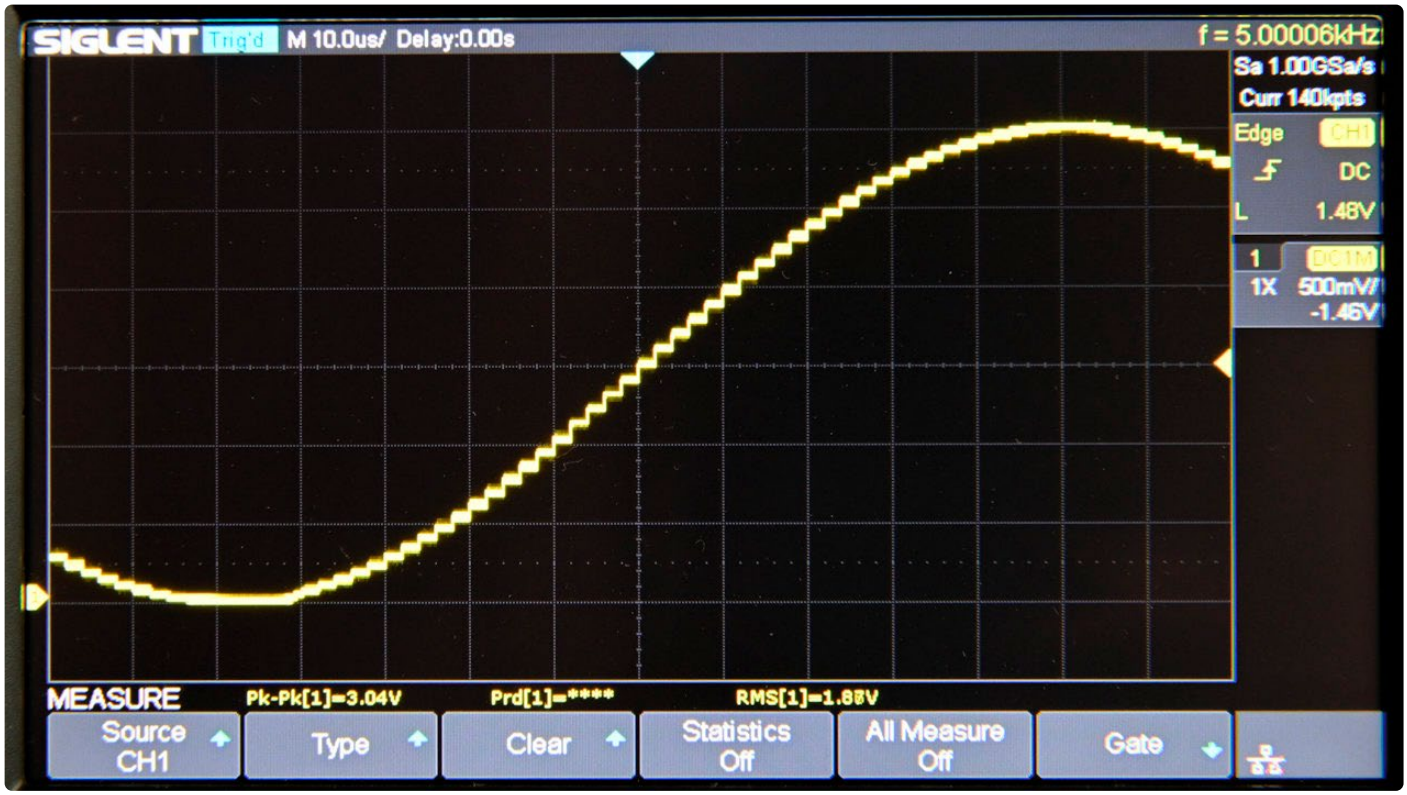


Bild 5. Die Signale des Funktionsgenerators sind recht grob und man kann die Quantisierungsschritte deutlich erkennen.

Ein gutes Geschäft

Das neue Fnirsi 2C53T [1] kostet zwar etwas mehr als das Vorgängermodell 2C23T, bietet aber klare Verbesserungen im Bereich des Oszilloskops. Besonders die Bandbreite von 50 MHz und die Abtastrate von 250 MSamples/s sind wichtige Neuerungen.
Auch wenn das Multimeter weitgehend gleich bleibt und der Frequenzbereich des Funktionsgenerators deutlich reduziert ist, würde ich das 2C53T allein wegen seiner Oszilloskop-Fähigkeiten wählen. Außerdem erhalten Sie zu diesem Preis auch zwei Oszilloskop-Tastköpfe und einen praktischen Aufbewahrungskoffer - ein netter Bonus!

Übersetzung: Mahy Arafa — 250137-02

Passende Produkte

- **FNIRSI 2C53T (3-in-1) 2-Kanal-Oszilloskop (50 MHz) + Multimeter + Signalgenerator**
  www.elektor.de/21112
- **FNIRSI 2C53P (3-in-1) 2-Kanal Oszilloskop (50 MHz) + Multimeter + Signalgenerator**
  www.elektor.de/20917
- **FNIRSI 2C23T (3-in-1) 2-Kanal-Oszilloskop (10 MHz) + Multimeter + Signalgenerator**
  www.elektor.de/20717

WEBLINKS

[1] Fnirsi 2C53T: https://www.elektor.de/products/fnirsi-2c53t-upgrade-3-in-1-2-ch-oscilloscope-50-mhz-multimeter-signal-generator
[2] Baggen, „Review: Fnirsi 2C23T Oszilloskop, Multimeter und Signalgenerator“, elektormagazine.de, April 2024: https://www.elektormagazine.de/review/fnirsi-2c23t-oszilloskop-multimeter-generator

Sie designen. Wir liefern.

Die neuesten Produkte für Ihre neuesten Designs.™

mouser.de/new

MOUSER ELECTRONICS