Mikrocontroller programmieren ohne PC – tastenprogrammierbare Steuerung TPS

Die Idee ist so einfach wie genial – ein kleiner Steuercomputer soll ohne PC oder Programmiergerät nur über ein paar Tasten programmiert werden. Dabei soll der Befehlssatz so kompakt sein, dass man ihn im Kopf behalten kann. Der Steuercomputer selbst sollte ebenso übersichtlich konfiguriert sein: 4 Eingänge, 4 Ausgänge, 1 PWM-Ausgang und 1 oder 2 ADC-Ports, fertig. Genau diese Idee ist nun in Form des neuen Franzis-Lernpakets „Mikrocontroller programmieren" zur Realität geworden.

Computerprogramm ganz einfach

Die Geschichte dieses Lernpakets war auf der Internet-Seite des bekannten Elektronik-Autors Burkhard Kainka [1] schon lange zu verfolgen. Im Urlaub kam ihm die Idee: Man müsste eine kleine Mikrocontroller-Platine dabeihaben, die mit Bordmitteln, ohne einen PC, programmierbar sein sollte. So könnte man z. B. auch im Urlaub sein Gehirn trainieren und etwa „eben mal, quasi am Strand" ein neues Geschicklichkeitsspiel für die Familie entwickeln. Zitat: „Ein kleiner Steuercomputer soll ohne PC oder Programmiergerät nur über ein paar Tasten programmiert werden können. Der Befehlsvorrat soll so einfach sein, dass man ihn im Kopf behalten kann, um notfalls ganz ohne Unterlagen ein Steuerprogramm zu entwickeln. Damit alles möglichst klein und überschaubar ist, soll es ein 4-Bit-System werden. Es gibt 4 digitale Ausgänge, 4 digitale Eingänge, intern verarbeitete Daten haben eine Breite von 4 Bit, und auch die Befehle sind nur mit 4 Bit kodiert, d. h., es gibt maximal 16 Befehle, die man sich merken muss." Als Anzeige sollten 4 binär anzeigende LEDs dienen, die abwechselnd Adressen, Befehle und Daten anzeigen sollen. Zum Programmieren und Starten des Programms sollten 3 Tasten reichen.

Die erste Version dieser tastenprogrammierbaren Steuerung (TPS) entstand mit einem unter BASCOM geschriebenen Interpreter auf einem Atmel-AVR. Später wurde das Programm mit einem C-Compiler auf den preiswerten Holtek-Controller HT46F47 portiert. Ein solcher wurde so schon beim „Modellbahn-Universal-Beleuchtungs-Set" von Franzis eingesetzt. Im neuen Lernpaket „Mikrocontroller programmieren" kommt eine auf einem IC-Sockel steckbare DIL-Version zum Einsatz, die den Vorteil hat,

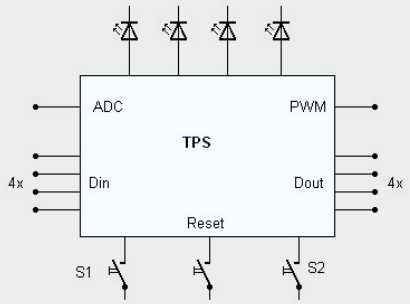


Bild 1: Die Grundidee der TPS: Blackbox-Controller mit 4 Eingängen, 4 Ausgängen, 3 Tastern, Analogport und PWM-Ausgang. Grafik: B. Kainka

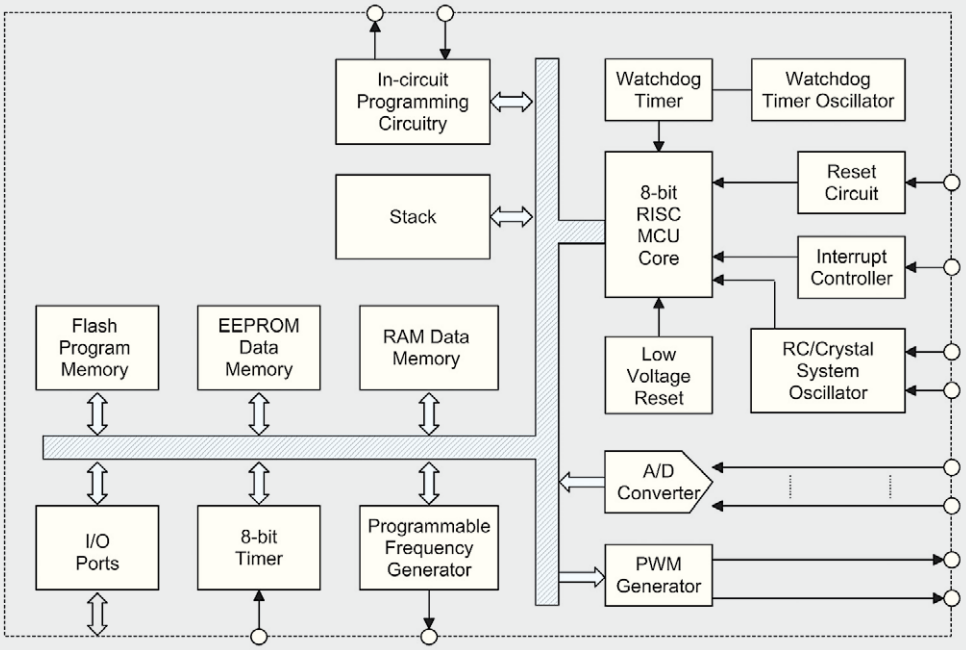


Bild 2: Der Aufbau des verwendeten Holtek-Controllers – alles drin, was ein vollwertiger Mikrocontroller benötigt. Grafik: Holtek

dass man direkt Anwendungen auf einen mit der Firmware versehenen Controller programmieren und diesen in einer eigenen Applikationsumgebung einsetzen kann. Dass diese interessante Lösung nicht nur „Programmiermuffel" ansprechen kann, sondern auch die Gemeinde der sonst auf AVRs Programmierenden herausfordert, kann man aktuell beim ebenfalls unter [1] laufenden Programmierwettbewerb erleben. Tatsächlich ist dieser kleine Steuercomputer etwas für die, die sich nicht mit der kompletten Programmierung von Mikrocontrollern herumschlagen möchten. Da setzt die TPS eben noch weit unterhalb der Arduino-Ebene an, wo man immerhin noch eine einfache, C-ähnliche Hochsprache erlernen muss.

Für den TPS-Controller genügen ganze 14 Befehle, die man nach kurzer Zeit im Kopf hat, und die tatsächlich nur über die 2 Tasten auf der Platine eingegeben und durch eine 3. Taste gestartet werden. Die Ausgabe der Ergebnisse erfolgt über 4 LEDs und 1 PWM-Ausgang. Bild 1 zeigt diese Grundidee. Statt der LEDs kann man genauso gut über eine Transistorstufe Relais anschließen und damit einen Aktor realisieren, z. B. für einen Zeitschalter, einen Dämmerungsschalter, eine Zufallssteuerung und andere Ideen. Auch analoge Vorgänge sind auswertbar, dazu stehen 2 Analog-Ports zur Verfügung. Hier kann der kleine Prozessor schon mehr als die Maschine in meiner lange zurückliegenden Studienzeit, an die ich zurückdenken musste: ein kleines Tastenpult, in das Rechenkombinationen in Befehlsform, Adresse für Adresse, Bit für Bit, einzugeben waren. Nach dem Speichern (damals noch auf Magnetbänder und Kernspeicher) konnte das Programm gestartet werden, und die Rechenergebnisse erschienen in kryptischer Form auf ein paar Leuchtanzeigen – ein Display hatte dieser Rechner noch nicht. Das Programmieren war freilich deutlich komplizierter als bei unserem kleinen Controller hier, aber der grundsätzliche Vorgang ähnlich.

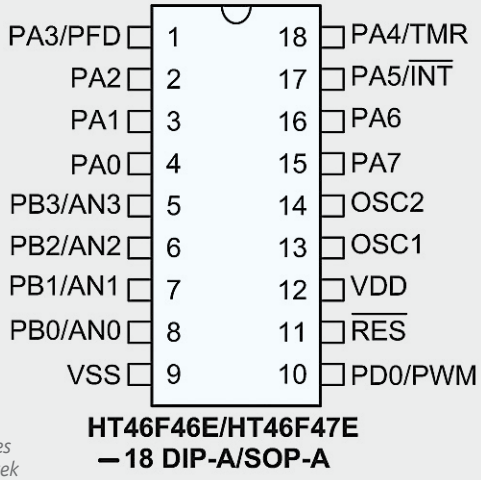


Bild 3: Anschlussbelegung des Holtek HT46F47. Grafik: Holtek

Der Controller

Die Wahl fiel auf den Holtek-Controller HT46F47, einen sehr preiswerten Mikrocontroller, den man in vielen Consumer-Geräten, von der Fernbedienung bis zum Küchenradio, als Steuerprozessor findet. Seine Struktur (Bild 2) ähnelt prinzipiell der eines AVR-Controllers, die Leistungsfähigkeit ist im Bereich der Tiny-AVRs bis hin zum AtMega8 einzuordnen.
Die Lieferung kann in den verschiedensten Gehäuse- und Port-Ausstattungsversionen erfolgen, hier kommt das 18-polige DIL-Gehäuse (Bild 3) zum Einsatz. Da auch dieser Controller einen internen RC-Taktoszillator enthält, ist die erforderliche Außenbeschaltung extrem minimal, wie an der Grundschaltung der Franzis-Experimentierplatine in Bild 4 gezeigt. Lediglich wenige Bauteile am Takt- und Reset-Eingang – das ist alles. Der Betrieb kann mit anwenderfreundlichen 2,2–5,5 V erfolgen, typischerweise wird das Lernpaket mit den obligaten 3 Mignonzellen, sprich 4,5 V, betrieben.

Der Controller bietet insgesamt 13 I/O-Ports an, verfügt über 1 Timer, 4 Analog-Ports (2 davon werden beim Lernpaket-Controller als Eingänge für die Programmierung benutzt) und einen PWM-Ausgang. Das ist auch schon fast alles, was der TPS-Anwender vom Controller wissen

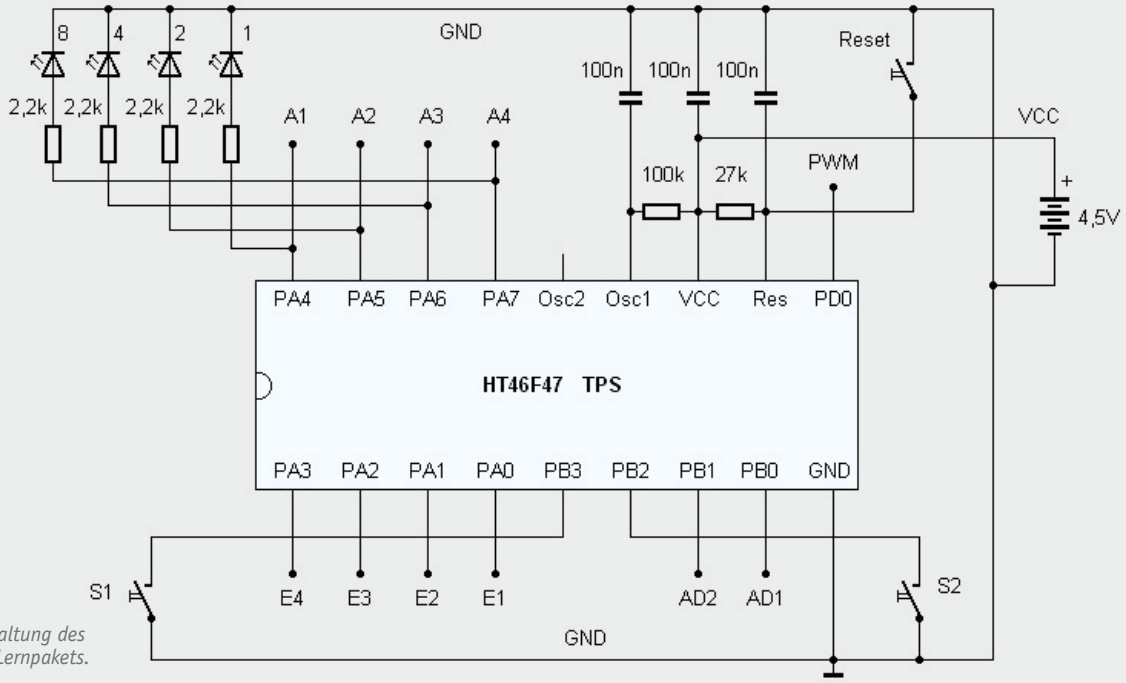


Bild 4: Minimal – die Schaltung des Experimentierboards des Lernpakets. Grafik: B. Kainka

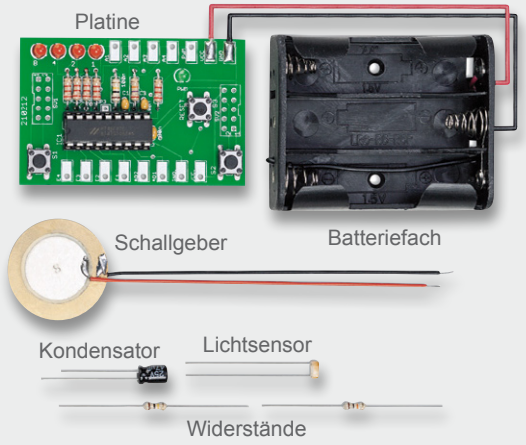


Bild 5: Gewohnt komplett – die gelieferten Bauteile des Lernpakets. Die Platine muss selbst bestückt werden.

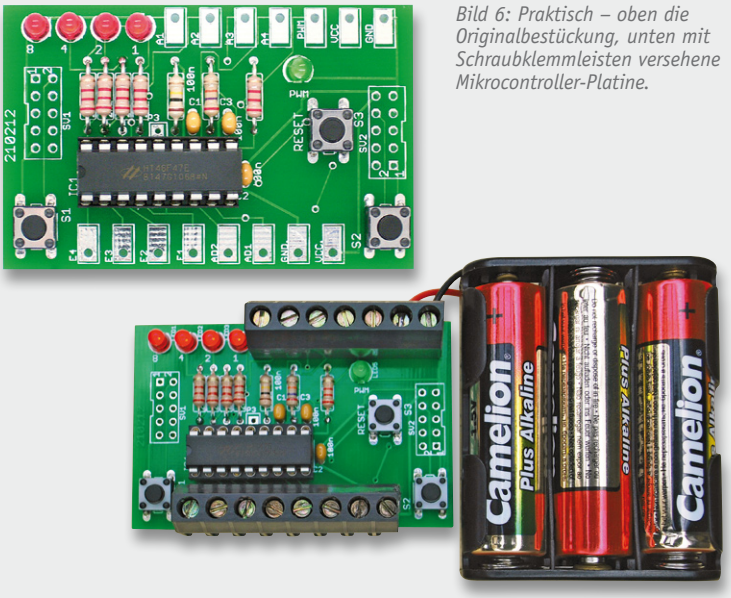


Bild 6: Praktisch – oben die Originalbestückung, unten mit Schraubklemmleisten versehene Mikrocontroller-Platine.

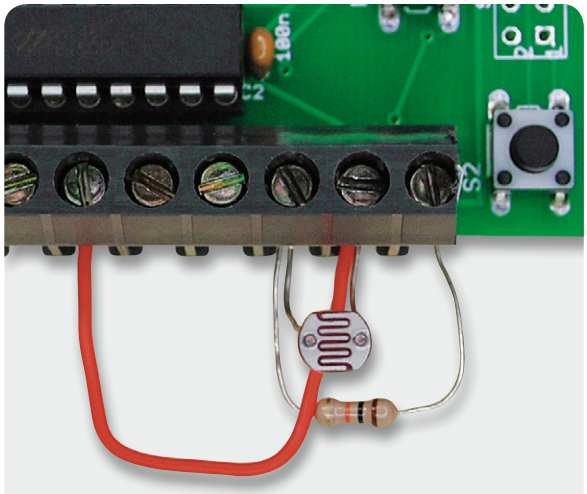


Bild 7: Experimente ohne Löten – direkt per Schraubklemme verbaute Zusatzbauteile.

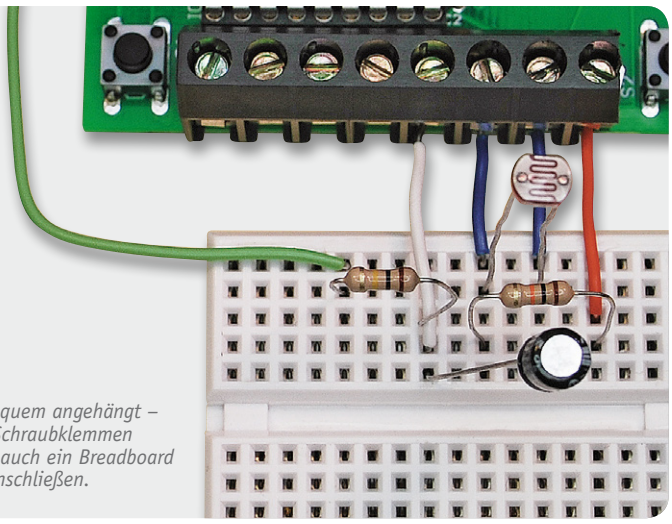


Bild 8: Bequem angehängt – über die Schraubklemmen lässt sich auch ein Breadboard einfach anschließen.

muss. Denn er schreibt im Grunde genommen nur via bereits installierten Interpreter direkt in den EEPROM des Controllers, von wo aus der Interpreter nach einem Reset das jeweilige Programm lädt und startet. Es sind auch mehrere kleinere Programme im EEPROM speicherbar, die, auf entsprechenden Adressen abgelegt und per aktiviertem I/O-Port ausgewählt, einzeln startbar sind.

So sind im Controller des Lernpakets bereits 4 Programme gespeichert, die sofort nutzbar sind und an denen sich mithilfe der ausführlichen Handbuchbeschreibung perfekt lernen lässt. Im 128 Byte fassenden EEPROM des Controllers können bis zu 128 TPS-Befehle untergebracht werden, sie sind nibbleweise als 8-Bit-Zahl organisiert (höherwertiges Nibble: Befehl, niederwertiges Nibble: Daten).

Das Lernpaket

Das sehr preiswerte Paket kommt in guter Franzis-Manier in einem stabilen Karton, mit gedrucktem Begleitbuch und als selbst zusammenzulötender Bausatz nebst einigen zusätzlichen, für die 20 Experimente des Begleitbuchs erforderlichen Bauteilen wie Fotowiderstand oder Piezo-Schallgeber (Bild 5).

Die Platine ist schnell aufgebaut, wir griffen gleich den Vorschlag des Begleitbuchs auf und haben unsere Platine mit passend zurechtgeschnittenen Schraubklemmleisten versehen (Bild 6). So kann man die wenigen externen Bauteile direkt, ohne löten zu müssen, in den Schraubklemmen verschrauben (Bild 7) oder besonders bequem auf einem Breadboard anordnen (Bild 8). Zusätzlich sind alle Anschlüsse auf 2 10-polige Anschlüsse für Wannenstecker geführt, so kann man Peripherie auch einfach per Flachbandkabel anschließen.

Bereits die intern „ab Werk" programmierten Beispielprogramme sind so gut erläutert, dass sich Programmstruktur und die Binäranzeige der LEDs schnell erschließen. Wie schnell sich übrigens 16 Binärzahlen lernen lassen, bewies der erste spontane Einsatz der Platine. Ich hatte sie am Nachmittag aufgebaut und am Abend den Kindern unserer Gäste präsentiert. Innerhalb von Minuten hatten die Kleinen (8 und 10 Jahre) beim Reaktionstest (Impulslängenmessung) sowohl die Binärzahlen als auch die Umrechnung in Millisekunden drauf. Und sie waren flink – schnellste Reaktionszeit: 30 ms.

Programmieren in wenigen Schritten

Ab Seite 31 des Begleitbuchs wird es interessant, das Programmieren beginnt. Zunächst wird gezeigt, wie man gespeicherte Programme über die LED-Anzeige ausliest. Nacheinander werden per Taste erst die Adresse, dann der Befehl, dann die zugehörigen Daten ausgelesen.

Sodann geht es sofort an das Programmieren des ersten Programms, es soll eine der LEDs einschalten und ist ganze 2 Byte lang. Das wird detailliert bis hin

Tabelle 2

Beispielprogramm Wechselblinker

Adresse	Befehl	Daten	Kommentar
00	1	1	LED1 ein
01	2	8	Warte 500 ms
02	1	8	LED8 ein
03	2	8	Warte 500 ms
04	3	4	Spring zurück auf Adresse 00

Tabelle 1

Befehle, Unterfunktionen und Parameter der TPS

Befehl	1	2	3	4	5	6	7	8	9	A	B	C	D	E
Daten	Port=	Wait	Jump-	A=	...=A	A=...	A=...	Page	Jump	C*	D*	Skip if ...	Call	Ret
0	0	1 ms	-0	0				0	+0	0	0		0	
1	1	2 ms	-1	1	B=A	A=B	A=A+1	1	+1	1	1	A>B	1	
2	2	5 ms	-2	2	C=A	A=C	A=A–1	2	+2	2	2	A<B	2	
3	3	10 ms	-3	3	D=A	A=D	A=A+B	3	+3	3	3	A=B	3	
4	4	20 ms	-4	4	Dout=A	A=Din	A=A–B	4	+4	4	4	Din.0=1	4	
5	5	50 ms	-5	5	Dout.0=A.0	A=Din.0	A=A*B	5	+5	5	5	Din.1=1	5	
6	6	100 ms	-6	6	Dout.1=A.0	A=Din.1	A=A/B	6	+6	6	6	Din.2=1	6	
7	7	200 ms	-7	7	Dout.2=A.0	A=Din.2	A=A And B	7	+7	7	7	Din.3=1	7	
8	8	500 ms	-8	8	Dout.3=A.0	A=Din.3	A=A Or B		+8	8	8	Din.0=0	8	
9	9	1 s	-9	9	PWM=A	A=AD1	A=A Xor B		+9	9	9	Din.1=0	9	
A	10	2 s	-10	10		A=AD2	A=Not A		+A	A	A	Din.2=0	A	
B	11	5 s	-11	11					+B	B	B	Din.3=0	B	
C	12	10 s	-12	12					+C	C	C	S1=0	C	
D	13	20 s	-13	13					+D	D	D	S2=0	D	
E	14	30 s	-14	14					+E	E	E	S1=1	E	
F	15	60 s	-15	15					+F	F	F	S2=1	F	

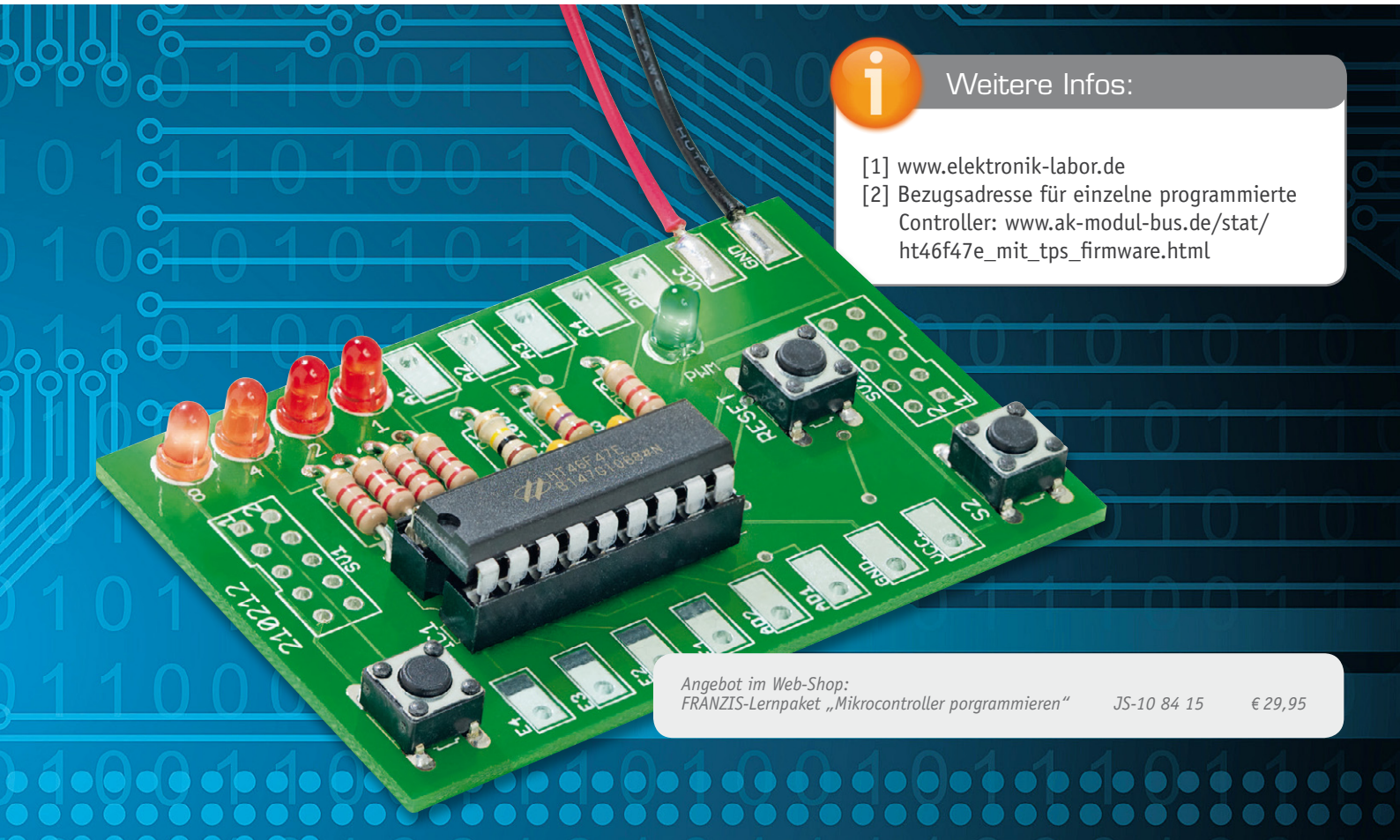
zur erforderlichen Anzahl der Tastendrücke von S1, S2 und Reset erklärt. Jetzt hat man die Bedienung „drauf", fehlen nur noch die Befehle. Das kriegen wir im nächsten Kapitel – ganze 14 Befehle gilt es, sich einzuprägen! Die meisten sind mit einem Parameter in Form einer 4-Bit-Zahl einzugeben, andere mit Unterfunktionen in Parameterform. Das können pro Befehl bis zu 16 Unterfunktionen sein. Klingt verwirrend, ist aber anhand der Befehlstabelle (Tabelle 1) schnell gelernt. Zudem gibt das Begleitbuch einen ausführlichen Exkurs zu allen Befehlen und Funktionen. Man muss sich nun nicht gleich alle Befehle samt Parameter einprägen, zunächst hilft die Befehlstabelle und im Verlauf der danach ebenso ausführlich besprochenen Programmbeispiele lernt man die Programmstruktur und quasi nebenbei die Anwendung von Befehlen und Parametern.

Anhand eines ganz einfachen Beispiels, eines kleinen, nur aus 5 Zeilen bestehenden Wechselblinker-Programms (Tabelle 2), kann man sich hier einmal beispielhaft die Handhabung der Befehlstabelle erarbeiten. Die in Tabelle 2 dargestellte Art der Programmtabelle ist der Standard bei der Programm-Erarbeitung, für sie gibt es innerhalb des Programmierwettbewerbs in [1] sogar eine Vorlage zum Ausdrucken.

Zurück zum Beispiel: Zuerst soll nach Anwahl der Adresse 00 die LED1 (diese ist an A1 angeschlossen) eingeschaltet werden. Der Befehl (siehe Kopfzeile der Befehlstabelle) dazu lautet „1" (Port aktivieren), die zugehörigen Daten (erste Spalte der Tabelle) lauten für den Port 1 „1". Nach Fortschalten auf Adresse 01 kommt nun die Eingabe der Wartezeit für das Umschalten der LEDs: Wir wählen den Befehl „Wait" (Befehl 2) und als Parameter 500 ms, also „8". Nach Ablauf der Wartezeit soll LED8 ebenfalls für 500 ms eingeschaltet werden, wobei LED1 wieder erlöschen soll. Also folgt auf Adresse 02 wieder Befehl 1, aber dieses Mal mit dem Parameter „8". Der folgende Wait-Befehl ist identisch mit dem ersten Wait-Befehl. Damit das Ganze nicht nur 1-mal durchläuft und dann stehen bleibt, wird auf Adresse 04 schließlich ein Rücksprungbefehl (Jump-) „3" zur Adresse 00, also 4 zurück („4") eingegeben. Jetzt haben wir eine Programmschleife produziert, die erst durch Reset abgebrochen werden kann.

Wie man an den Parametern und Unterbefehlen erkennen kann, erlauben die wenigen Befehle dennoch recht leistungsfähige Programmabläufe bis hin zu Regelschleifen und komplexere Programme. Auch solche Anwendungen beschreibt das Begleitbuch, das im Anhang gleich noch einige öfter benötigte Unterprogramme auf einen Blick zusammenfasst. Was hier möglich ist, zeigen die Einsendungen zum bereits erwähnten Programmierwettbewerb: LiPo-Ladegerät, Lüftersteuerung, Dimmer, Reaktionstester waren bereits nach wenigen Tagen die ersten veröffentlichten Lösungen.

Unter dem Strich lässt sich resümieren, dass die Idee dieses Lernpakets wirklich faszinierend ist: Man muss für kleinere Mikrocontroller-Lösungen bzw. für den Ersatz von „IC-Gräbern" nicht unbedingt eine komplette Programmiersprache lernen, keinen PC und keinen speziellen Programmer zum Programmieren des Chips bemühen. Einfach einen bei [2] zu beziehenden Chip mit der Firmware aufstecken, das zuvor aufgeschriebene Programm eingeben – fertig und ab in die Applikationsschaltung! Wetten, dass diese Art der „Einstiegsdroge" unweigerlich dazu führt, sich doch einmal an eine Hochsprache zur Mikrocontroller-Programmierung zu wagen? Die Grundlage in Form dieses Lernpakets ist jedenfalls eine sehr gelungene Voraussetzung dafür. ELV



Weitere Infos:

[1] www.elektronik-labor.de
[2] Bezugsadresse für einzelne programmierte Controller: www.ak-modul-bus.de/stat/ht46f47e_mit_tps_firmware.html