Vorgestellt: Franzis-Lernpaket Maker Kit Controller Board

JÖRG BISCHOF – DM6RAC

Für Funkamateure und Hobbyelektroniker, die sich mit dem Thema Hard- und Software beschäftigen möchten, ist dabei seltener der „große" Computer von Interesse, sondern eher der Mikrocontroller. Letzterer hat den Vorteil, dass sich trotz überschaubaren Hardwareaufwands recht vielseitige und anspruchsvolle Projekte realisieren lassen. Um den Einstieg zu erleichtern, gibt es eine ganze Reihe interessanter Produkte. Eines davon ist das Maker Kit des Franzis-Verlags.

Wer sich bisher noch nicht mit Hard- und Software beschäftigt hat und in das Thema einsteigen möchte, steht zunächst vor der Frage: Wie sollte ich anfangen? Eine erste Hürde stellt in der Regel die Programmiersprache dar.

Bild 1: Der Franzis-Verlag liefert den Bausatz *Maker Kit Controller Board* in einer optisch ansprechenden Verpackung. Werkfoto

Als ich selbst vor vielen Jahren begann, mich mit Mikrocontrollern zu beschäftigen, konnte ich anfangs nicht so recht verstehen, wie einfache Zahlen und Anweisungen an der Hardware etwas steuern können. Damals programmierte ich in *Assembler* und setzte die Befehle in die Maschinensprache um. Dies ist zwar umständlicher als mit einer Hochsprache wie z. B. *C/C++* oder *Python*, hat aber den Vorteil, dass man „hautnah" an den einzelnen Pins des Controllers ist und relativ schnell die Wirkungsweise der Befehle versteht.

Maker Kit vom Franzis-Verlag

Hier setzt das im Folgenden vorgestellte *Maker Kit Controller Board* an, Bild 1. Es ist u. a. bei *www.franzis.de* erhältlich und enthält alle benötigten Komponenten, von der Platine mit den dazugehörigen Bauelementen über einige zusätzliche Bauteile für Experimente bis hin zum Anleitungsbüchlein. Zusätzlich werden nur noch drei AA-Batterien benötigt.

Der verwendete 8-Bit-Mikrocontroller ist ein *HT46F47E* des Herstellers *Holtek* aus Taiwan [1]. Dieser im Amateurbereich wenig gebräuchliche Typ ähnelt den *AT-Tiny*-Mikrocontrollern des Herstellers *Microchip*.

Der *HT46F47E* besitzt einen Programmspeicher (engl. *flash memory*) in der Größe 2K × 14 Bit und darüber hinaus 64 KB RAM und 128 KB EEPROM. Hinzu kommen 13 Ein-/Ausgangsanschlüsse, vier Eingänge für einen A/D-Umsetzer mit 9 Bit Auflösung sowie ein 8-Bit-Zähler/Zeitgeber (engl. *timer*). Einem Einsteiger sagen diese Angaben vielleicht noch nicht viel. Sie sind aber zur Beurteilung der Leistungsfähigkeit des Mikrocontrollers von Interesse.

Das *Maker Kit* verfolgt den Ansatz einer sogenannten *tastenprogrammierbaren Steuerung* (TPS). In diesem Fall werden die Maschinenbefehle in Form von Dualzahlen direkt eingegeben. Um das Ganze überschaubar zu halten, verfügt das Kit über vier Ausgänge, abgeschlossen mit roten LEDs, mit maximal 10 mA belastbar, sechs Eingänge und drei Taster, zwei davon für die Befehlseingabe.

Durch die bereits im Mikrocontroller vorhandene Firmware kommt man mit nur 14 Programmierbefehlen aus. Dies hört sich zunächst wenig an, reicht aber aus, um zahlreiche Programmierexperimente durchzuführen und dadurch die Wirkungsweise des Mikrocontrollers verstehen zu lernen.

Bestückung der Platine

Auf die Platine sind aufzulöten: eine IC-Steckfassung, sieben Widerstände, fünf LEDs, drei Scheibenkondensatoren und drei Taster. Ich empfehle, mit den Widerständen zu beginnen. Kleiner Tipp: Damit die LEDs und Kondensatoren nicht aus den Bohrungen herausrutschen, einen Anschlussdraht nach dem Durchstecken einfach umbiegen, den anderen anlöten und dann den umgebogenen wieder geradebiegen und ebenfalls verlöten.

Bei den LEDs ist auf die richtige Polarität zu achten. Der längere Draht ist die Anode. Da die Katoden miteinander verbunden und an Masse liegen, kommt also der kurze Anschluss in Richtung der äußeren Kante. Das Ergebnis ist in Bild 2 zu sehen.

Nach dem Einsetzen des Mikrocontrollers in die IC-Fassung (Position der Kerbe am IC-Gehäuse beachten!) und dem Anschluss der Betriebsspannung startet das erste Programm automatisch. Es handelt sich um einen Wechselblinker.

Verbindet man E1 mit Masse und drückt die Reset-Taste, startet ein zweites Anzeigeprogramm. Die roten LEDs geben nun das Ergebnis eines Dualzählers aus und die grüne wird heller.

Programmierung

Im Begleitheft tauchen zu den Beispielen von Anfang an die Programm-Quelltexte (engl. *listings*) auf. Ich finde, dass die Beschreibung für Anfänger dadurch schwieriger zu verstehen ist. Wenn ich die Programmierung nachladen möchte, bekomme ich z. B. nicht die des Wechselblinkers zu sehen, der als erstes Programm läuft, sondern den Grundzustand (S. 46, Listing 3.1).

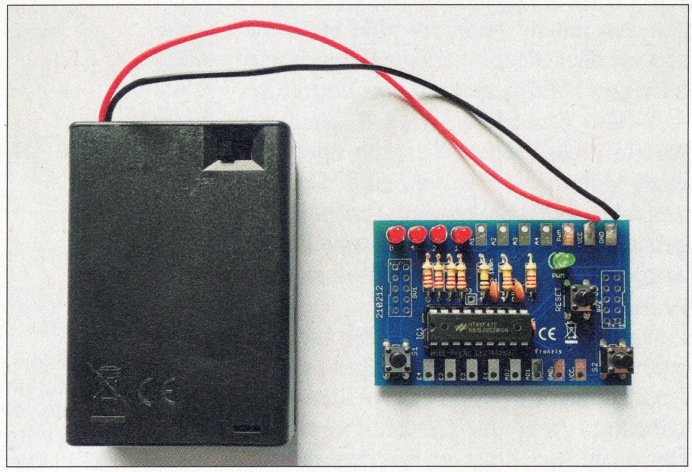


Bild 2: Vollständig bestückte Platine der Hardware des Bausatzes; das schwarze Kästchen enthält drei AA-Batterien zur Spannungsversorgung mit 4,5 V. Es gehört nicht zum Lieferumfang. Foto: DM6RAC

Ergänzte Befehls- und Datenliste zur Erstellung von Programmen für das *Maker Kit Controller Board* gemäß Begleitheft zum Bausatz

Befehle →	1 PORT=	2 Wait	3 Jump -	4 A=	5 ...=A	6 A=...	7 A=...	8 Page	9 Jump	A C*	B D*	C Skip if ...	D Call	E Ret
Daten 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	

Dies liegt an der Art, wie die Firmware programmiert wurde. Wir haben es hier mit zwei verschiedenen Festwertspeichern zu tun. Die Firmware, der dieser spezielle Befehlssatz zugrunde liegt, wurde in den Flash-Speicher programmiert.

Hier liegen auch die beiden Beispielprogramme. An den Flash-Speicher kommt der Anwender nicht heran. Die vom Anwender erstellten Programme liegen im EEPROM. Beim Start kontrolliert die Firmware, ob sich ein Programm im EEPROM befindet. Wenn ja, wird dieses ausgeführt, anderenfalls eines der Beispielprogramme.

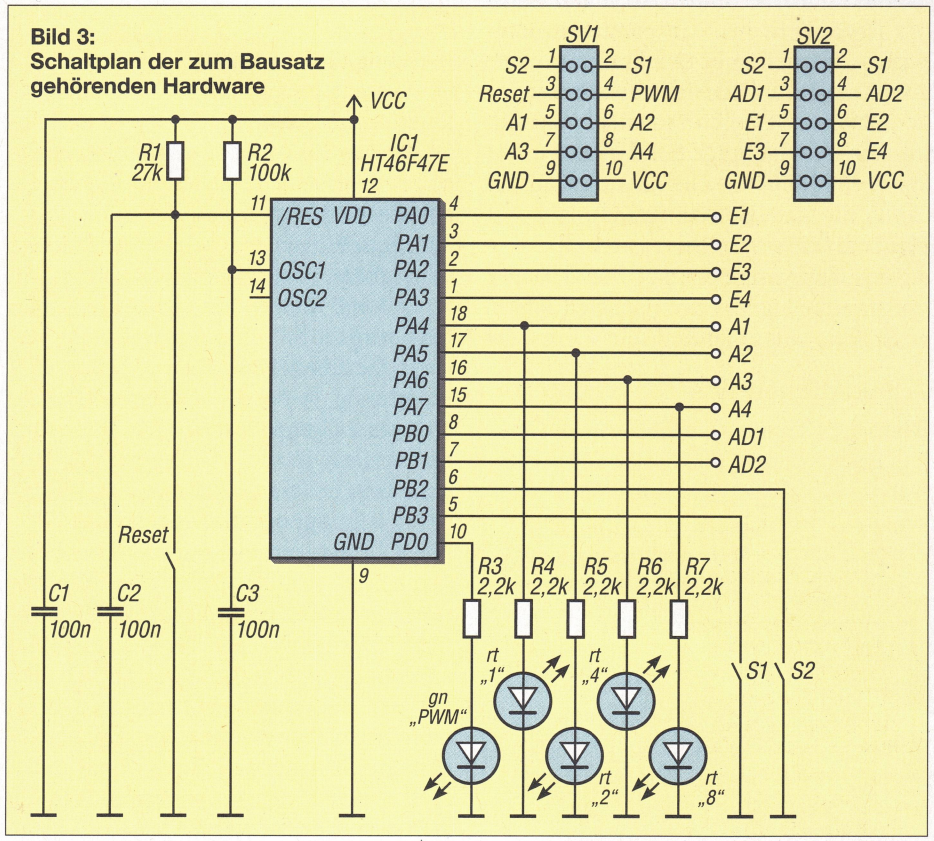
Die Befehlseingabe erfolgt über die Taster S1 und S2, Bild 3. Wie bereits erwähnt, stehen 14 Befehle zur Verfügung, hexadezimal mit den Nummern *0x1* bis *0xE* versehen und in der Tabelle einfach nur mit *1...E* bezeichnet. Die Umrechnung von Dezimal- in Binär- bzw. Hexadezimalzahlen sollte dem Anwender geläufig sein. Auf S. 50 des Begleithefts wird diesbezüglich auf das Visual-Basic-Programm *TPS-Tippomat* verwiesen. Bei mir hat es allerdings unter *Windows 11* die Funktion verweigert. Dies lag vielleicht daran, dass Virtual Basic inzwischen schon recht alt ist.

Um in den Programmiermodus zu gelangen, drückt man den *Reset*-Taster und S2 gleichzeitig, lässt zuerst *Reset* los und eine halbe Sekunde später S2. Letzterer ermöglicht es nun, durch das Programm zu scrollen. Jede Programmzeile besteht aus Adresse (wird nicht eingegeben), Befehl und Daten.

Die Anzeige erfolgt binär durch die vier roten LEDs, siehe S. 46 der Anleitung. Mithilfe der Taste S1 können Befehle oder Daten geändert oder eingegeben werden, siehe Tabelle. Mit S2 geht man an die Stelle, die zu ändern ist (Befehl oder Daten) und gibt die Werte mit S1 ein. S1 ist dabei so oft zu drücken, wie die Nummer laut Tabelle vorgibt. Aber aufpassen: Die Nummerierung beginnt mit *0* und diese Zahl erhält man auch beim ersten Druck auf S1. Um die Zahl *3* einzugeben, ist S1 also viermal zu drücken. Mittels S2 erfolgt dann die Bestätigung.

Die Befehle selbst werden, zusammen mit Beispielen, ab S. 52 beschrieben. Der Befehlsliste, die an zwei Stellen im Begleitheft aufgeführt ist, fehlt leider die Beschriftung der Achsen für Befehle und Daten. Ich habe diese daher in der Tabelle vervollständigt.

Bild 3:
Schaltplan der zum Bausatz gehörenden Hardware



■ Fazit

Trotz der etwas gewöhnungsbedürftigen Programmierung kann der Bausatz dem Einsteiger helfen, zu durchschauen, wie ein Mikrocontroller arbeitet. Im Gegensatz zu höheren Programmiersprachen muss nämlich hier jede Operation separat betrachtet werden. Der Vorteil ist, dass man das Verhalten genau abbilden und besser nachvollziehen kann.

Ich finde, dass die Arbeit auf der untersten Programmierebene in dieser Hinsicht wirklich hilfreich ist. Später, mit dem Einsatz höherer Sprachen, werden dann viele Schritte zu einem zusammengefasst. Wer die Grundlagen verstanden hat, ist dann klar im Vorteil.

dm6rac@darc.de

Literatur

[1] Holtek: Datenblatt HT46F47E. *www.holtek.com/documents/10179/116711/HT46F46E_47E_48E_49Ev180.pdf*