CE

Tastenprogrammierbare Steuerung

Mikrocontroller findet man überall: in Haushaltsgeräten, in Geräten der Unterhaltungselektronik, in Fahrzeugen, in Messgeräten und sogar in unbemannten Raumfahrzeugen. Überall tun sie Dinge, die ihnen ein Programm befiehlt. Es ist spannend, auch selbst einmal einfache Steuerprogramme zu erzeugen.

Der erste Schritt ist immer, einen Mikrocontroller oder Prozessor auszusuchen, der möglichst genau zu der gewünschten Aufgabe passt. Man hat die Auswahl zwischen unzähligen Typen verschiedener Firmen. Und auch die Programmiersprache kann gewählt werden. Meist werden Assembler und C angeboten, in vielen Fällen auch Basic oder eine andere Hochsprache. Normalerweise braucht man zur Programmierung aufwendige Software und ein Programmiergerät. Neben dem finanziellen Aufwand ist auch die Einarbeitungszeit nicht zu vernachlässigen.

Der hier verwendete Mikrocontroller ist ganz anders. Zum Programmieren brauchen Sie nicht mehr als zwei Tastschalter. Die *Tastenprogrammierbare Steuerung* (TPS) kennt nur relativ wenige Befehle, die sich leicht erlernen lassen und die mithilfe der Tasten in den Controller programmiert werden. Eine Änderung des Programms ist jederzeit und ohne besondere Hilfsmittel möglich.

Das System eignet sich besonders für kompakte Anwendungen im Bereich Messen, Steuern und Regeln. Viele Aufgaben sind mit diesem System bereits vollwertig lösbar. Dazu kommt, dass Sie den Mikrocontroller nach erfolgreicher Programmierung in eigene Schaltungen einbauen können. Grundwissen im Bereich Elektronik wird dabei vorausgesetzt.

Zugleich eignet sich das System auch als Grundlage für die Ausbildung und für die ersten Schritte in die Mikrocontroller-Programmierung. Erfolge stellen sich schneller ein als bei anderen Systemen. Die Strukturen sind aber ähnlich wie in anderen Programmiersprachen, sodass der spätere Übergang erleichtert wird.

Inhaltsverzeichnis

1	Einführung	5
2	Wechselblinker	8
3	Binärzähler und PWM-Ausgabe	9
4	Analog-Digital-Wandler	13
5	Zufallsgenerator	15
6	Impulslängenmessung	17
7	Programme auslesen	19
8	Programme eingeben	21
9	Wiederherstellung der Beispielprogramme	23
10	TPS-Grundbefehle	23
11	Rechnen mit Variablen	26
12	Sprünge und Verzweigungen	28
13	Befehlsübersicht	30
14	Zählschleifen	31
15	Vergleiche	32
16	AND, OR und XOR	33
17	Unterprogramme	34
18	Dämmerungsschalter	36
19	LED-Dimmer	36
20	Zahlenschloss	38
21	Anhang	40

1 Einführung

Das Prinzip des TPS-Controllers ist einfach. Man hat vier digitale Eingänge E1 bis E4 und vier digitale Ausgänge A1 bis A4. Außerdem gibt es zwei analoge Eingänge AD1 und AD2 sowie einen quasi-analogen PWM-Ausgang. Ein Reset-Eingang mit einer angeschlossenen Reset-Taste setzt ein Programm an den Anfang zurück. Der Controller wird mit drei AA-Zellen mit ca. 4,5 V versorgt und kann in einem Bereich von 2,2 V bis 5,5 V arbeiten.

Technische Daten:
Mikrocontroller: HT46F47
Taktfrequenz: 2 MHz
Internes EEPROM: 128 Bytes
Spannungsversorgung VCC: 2,2 V bis 5,5 V
Stromaufnahme: 1 mA bei 4,5 V
4 Ausgangs-Ports: belastbar bis 10 mA
1 PWM-Ausgang: belastbar bis 10 mA
4 Eingangs-Ports: Ruhezustand 1
2 analoge Eingänge: 0 V ... VCC
2 Tasteneingänge: Ruhezustand 1

Bauteile im Lernpaket:
Steckboard
Batteriefach 3 * AA
Draht
HT46F47 mit TPS-Firmware
3 Tastschalter
4 LEDs 5 mm, rot
1 LED 5 mm, grün
1 LDR
3 Scheibenkondensatoren 100 nF
1 Elko 47 µF
5 Widerstände 2,2 kΩ
1 Widerstand 10 kΩ
1 Widerstand 27 kΩ
2 Widerstände 100 kΩ

Zur Programmierung braucht man die beiden Tasten S1 und S2 und eine einfache LED-Anzeige aus vier LEDs an den Ausgängen A1 bis A4. Es gibt insgesamt 14 einfache Befehle mit zugehörigen Daten oder Unterbefehlen. Befehle und Daten sind jeweils als 4-Bit-Binärzahlen im Bereich 0000 bis 1111 (dezimal 0 bis 15) codiert und werden an den Anzeige-LEDs direkt sichtbar. Die jeweilige Zahl wird beim Programmieren durch Tastendrücke auf S1 programmiert. Mit S2 wird jeweils zwischen Befehl und Daten gewechselt und die Adresse der Befehlszeile erhöht. Die gesamte Programmstruktur ist so einfach, dass man sie nach einiger Übung aus dem Kopf beherrschen kann.

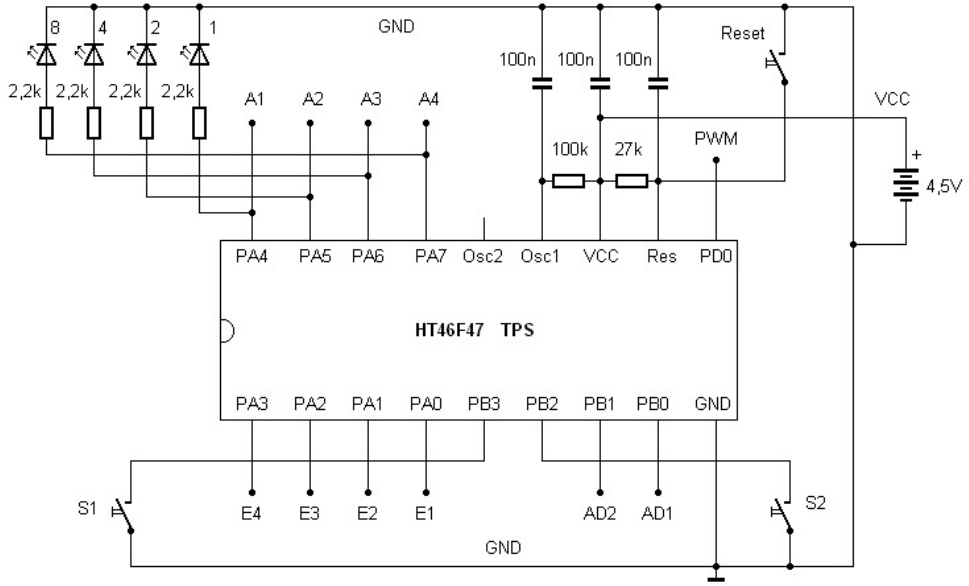

Abb. 1: Grundschaltung des Systems

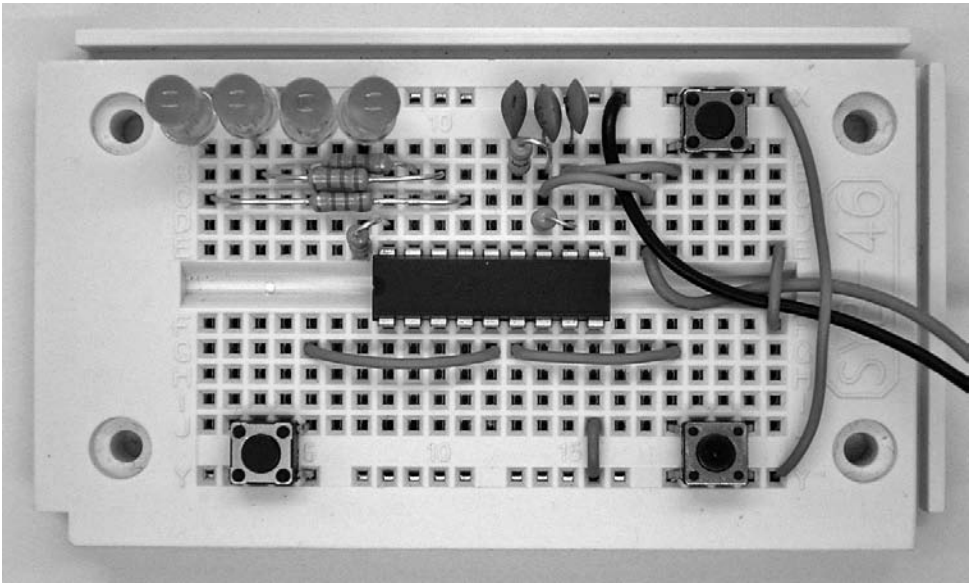

Abb. 2: Standardaufbau mit Tastschaltern

Einige Grundprogramme sind bereits im Auslieferungszustand (Default) im TPS-Controller vorhanden und können direkt gestartet werden. Deshalb ist es möglich, den Controller Schritt für Schritt in Betrieb zu nehmen. Machen Sie sich zunächst mit den Hardware-Funktionen vertraut und beginnen Sie dann erst mit eigenen Programmen.

Bei den ersten Tests werden kleine Programme gestartet, die bereits fertig im Controller vorliegen. Die zugehörigen Programmlisten vermitteln einen ersten Eindruck von den Möglichkeiten. Sie werden jeweils nur kurz erläutert. Die genaue Erklärung der einzelnen Befehle folgt im nächsten Kapitel.

Bauen Sie für den ersten Versuch nur die Grundkonfiguration mit dem Controller und den erforderlichen Zusatzelementen auf der Steckplatine auf. Unbedingt erforderlich sind:

- Anschluss der Spannungsversorgung an GND (Minus) und VCC (Plus)
- Ein Abblockkondensator 100 nF zwischen VCC und GND
- Reset-Widerstand nach VCC und Reset-Kondensator nach GND
- Oszillatorwiderstand 100 kΩ nach VCC und Kondensator nach GND

Der Mikrocontroller HT46F47 wird mit seinem internen RC-Oszillator betrieben. Der Widerstand am Osc1-Eingang legt die Taktfrequenz fest. Mit 100 kΩ wird eine Frequenz von ca. 2 MHz eingestellt. Bei Bedarf kann mit kleinerer oder größerer Geschwindigkeit gearbeitet werden. Der angeschlossene Kondensator dient nur zur Abblockung und hat keinen Einfluss auf die Taktfrequenz. Der Anschluss Osc2 bleibt frei. Bei Bedarf kann man aber hier einen zusätzlichen Widerstand gegen VCC anschließen und Impulse mit einem Viertel der Taktfrequenz auskoppeln.

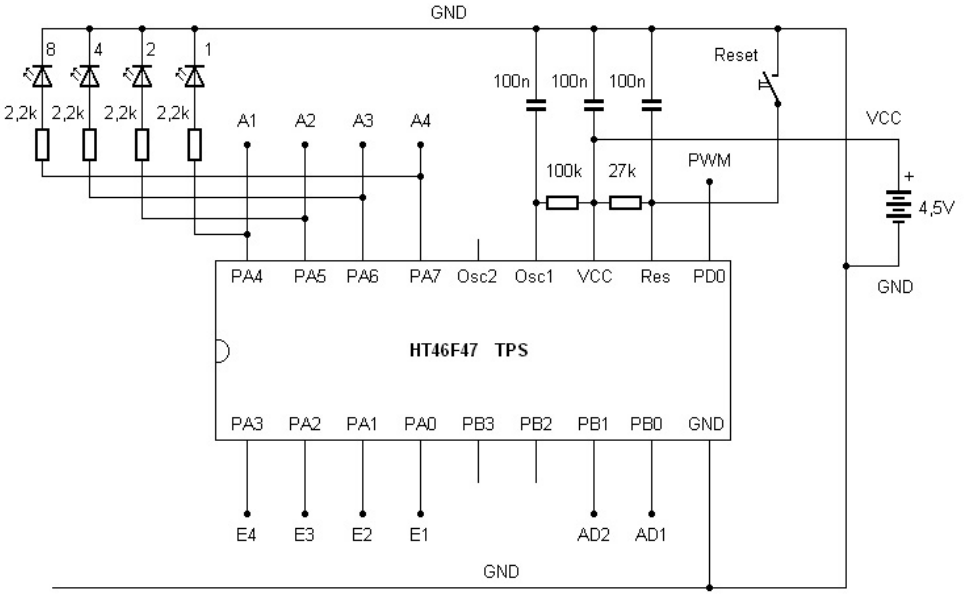

Abb. 3: Vier LEDs an den Ausgängen

Verwenden Sie die obere und die untere Versorgungsschiene auf der Steckplatine als Masseanschluss GND. Hier wird das schwarze Kabel des Batteriefachs, also der Minuspol, angeschlossen. Die Plus-Leitung VCC wird mit dem roten Anschlusskabel des Batteriefachs verbunden. Eine Falschpolung muss unbedingt vermieden werden, da sie den Controller zerstören kann. Bauen Sie ein kurzes Drahtstück als Zugentlastung ein. Die einmal angeschlossene Spannungsversorgung sollte möglichst immer verbunden bleiben. Nehmen Sie zum Abschalten eine der Batterien aus dem Fach.

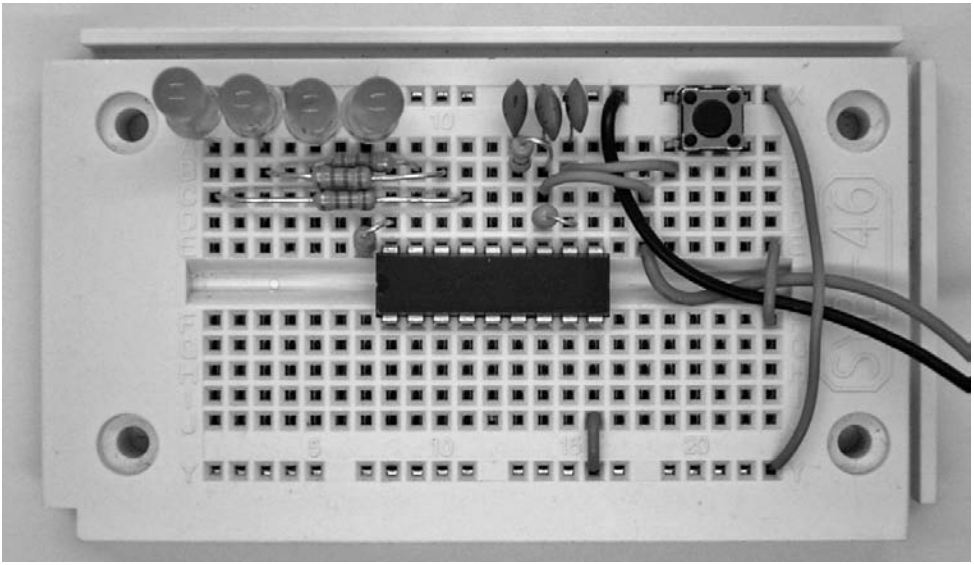

Abb. 4: Minimale Beschaltung mit LEDs

Setzen Sie zusätzlich bereits den Reset-Taster ein, und schließen Sie vier LEDs mit Vorwiderständen von 2,2 kΩ an. Diese werden für die ersten Hardware-Tests gebraucht. Achten Sie auf die Reihenfolge. A1 wird an die linke LED angeschlossen und A4 an die rechte. Damit hat man eine Binäranzeige mit dem höchstwertigen Bit links. Das ist vor allem bei der späteren Programmierung hilfreich.

2 Wechselblinker

Setzen Sie nun drei 1,5-V-Batterien oder alternativ drei NiMh-Akkus in das Batteriefach ein. Damit starten Sie das erste Beispielprogramm mit einem Wechselblinker mit der linken und der rechten LED. Die Blinkfrequenz beträgt etwa 1 Hz. Das Programm-Listing zeigt das einfache Programm mit nur fünf Zeilen. Es wird abwechselnd die LED 1 und die LED 8 eingeschaltet. Dazwischen liegen Wartebefehle mit einer Wartezeit von 0,5 s. Ein Rücksprung zum Anfang sorgt dafür, dass das Blinken endlos wiederholt wird. Die einzelnen Befehle werden weiter unten noch genauer erläutert. Sie können aber an diesem Beispiel schon die Einfachheit der Programmierung erkennen. Die Firmware des Controllers besitzt einen Interpreter, der die einfachen Kommandos erkennt und ausführt. Programme werden daher um ein Vielfaches kompakter als in anderen Systemen.

Das Beispiel belegt den Adressbereich ab 20h (dezimal 32). Mehrere Programme im oberen Adressbereich können später auch aus eigenen Anwendungen heraus gestartet werden. Die Adressen können alternativ auch mit eigenem Programmcode überschrieben werden. Bei Bedarf kann der Controller aber auch wieder in den Grundzustand zurückgesetzt werden, wobei die ursprünglichen Beispielprogramme wiederhergestellt werden.

Adresse	Befehl	Daten	Kommentar
20	1	1	LED 1
21	2	8	Warte 500 ms
22	1	8	LED 8
23	2	8	Warte 500 ms
24	3	4	Springe -4

Listing 1: Wechselblinker

Falls das gewünschte Ergebnis ausbleibt, überprüfen Sie zuerst die korrekte Polung der LEDs. Hilfreich ist auch eine Messung einiger Spannungen. Verwenden Sie z. B. ein Digitalmultimeter im 20-V-Bereich und lassen Sie den Minusanschluss an GND. Alle Spannungen werden damit gegen GND gemessen:

VCC: 4,5 V
Reset: 4,5 V
Osc1: 1,5 V
E1 bis E4: 4,5 V
A1: wechselnd
A2, A3: 0V
A3: wechselnd

3 Binärzähler und PWM-Ausgabe

Alle digitalen Eingänge sind mit einem internen Widerstand gegen VCC hochgezogen (Pullup-Widerstand) und haben eine Ruhespannung in Höhe der Betriebsspannung. Sie können aber jeden der Eingänge mit einem Draht oder einem Kontakt an GND legen. Beim Start liest das Default-Programm den Port-Zustand und wertet ihn aus. Einzelne Anschlüsse können an GND gelegt werden, sodass hier ein Nullzustand gelesen wird. In Abhängigkeit vom Ergebnis werden verschiedene Programme aufgerufen.

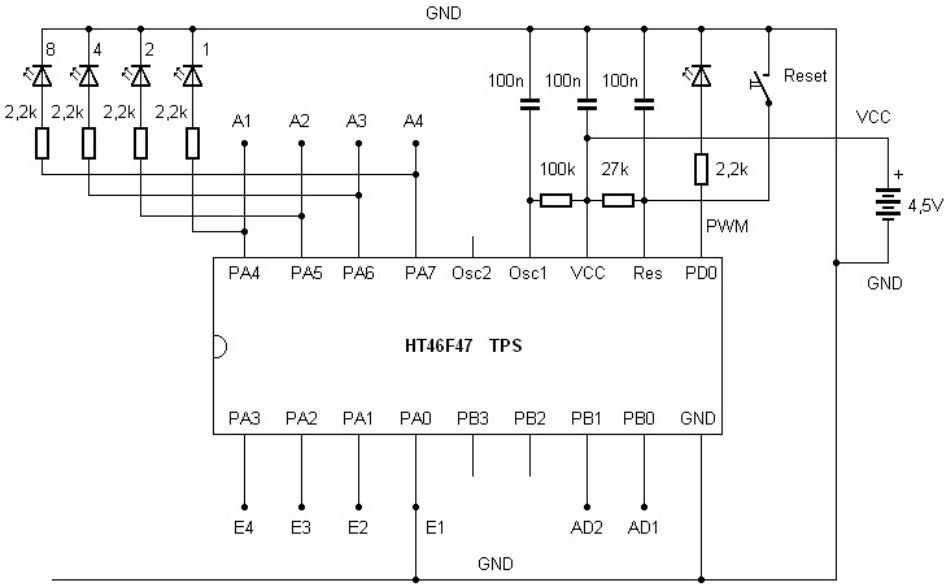

Abb. 5: Einsatz der PWM-LED

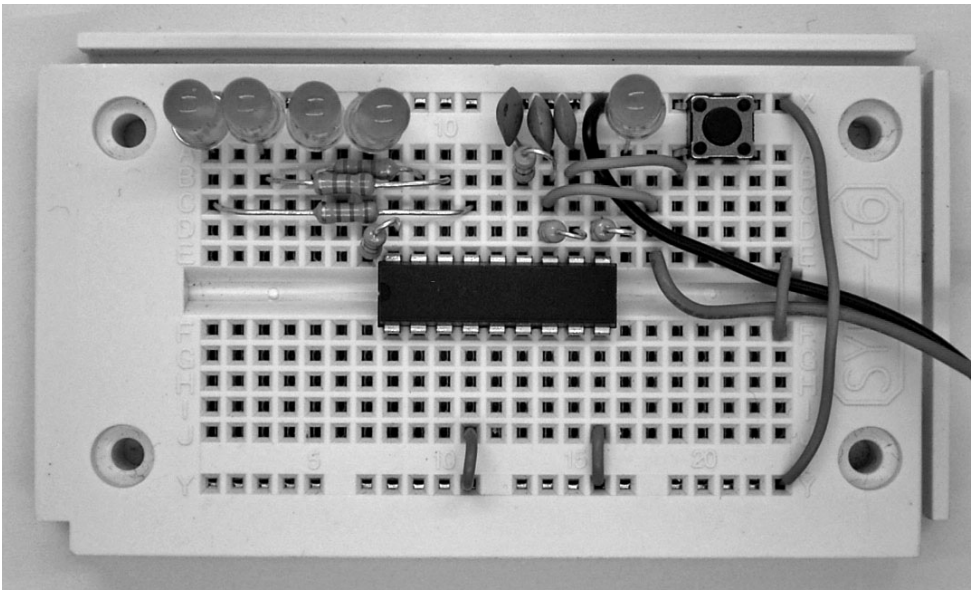

Abb. 6: Start des Binärzählers

Legen Sie E1 an GND. Damit startet nach einem Reset das zweite Beispielprogramm. Es zählt die Ausgangszustände binär hoch. Es werden laufend die Zustände 0000 (dezimal 0) bis 1111 (dezimal 15) durchlaufen. Das Programm verwendet die Variable A für eine einfache Addition und zur Ausgabe an die digitalen Ausgänge sowie an den PWM-Ausgang. Die Befehle 7 und 5 besitzen Unterfunktionen, die als Daten geschrieben werden.

Adresse	Befehl	Daten	Kommentar
25	7	1	A = A + 1
26	5	4	Port = A
27	5	9	PWM = A
28	2	6	Warte 100 ms
29	3	4	Springe -4

Listing 2: Binärzähler mit LED- und PWM-Ausgabe

Das Zählprogramm kann als Übungsprogramm für das Lesen von Binärzahlen eingesetzt werden, die für die eigene Programmierung beherrscht werden müssen. Jede der vier LEDs stellt ein Bit dar. Insgesamt kann daher eine 4-Bit-Zahl angezeigt werden. Die LEDs werden im Schaltplan entsprechend ihrer Wertigkeit 8, 4, 2 und 1 bezeichnet. Durch Addition der jeweiligen Werte erhält man die Dezimalzahl. In hexadezimaler Schreibweise werden die Zahlen 10 bis 15 durch die großen Buchstaben A bis F dargestellt.

8	4	2	1	Dezimal	Hexadezimal
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

Das Programm lässt sich auch als Blinkgeber für unterschiedliche Frequenzen einsetzen.

Der nächsthöhere Ausgang hat jeweils die halbe Frequenz oder die doppelte Periodendauer:

A1: 200 ms
A2: 400 ms
A3: 800 ms
A4: 1.600 ms

Zusätzlich werden die aufsteigenden Zahlenwerte auch an den PWM-Ausgang (Pulsweitenmodulation) ausgegeben. Das PWM-Signal ist ein Rechtecksignal mit einer Frequenz von ca. 16 kHz. Die Pulslänge wird dabei gesteuert, sodass das Puls-Pausen-Verhältnis die durchschnittliche Einschaltdauer und damit die Helligkeit der LED bestimmt. Die Helligkeit der hier angeschlossenen LED wird in 15 Stufen zwischen Null und voller Helligkeit gesteuert.

Ein PWM-Signal kann mithilfe eines RC-Tiefpassfilters zu einer Gleichspannung geglättet werden. Der PWM-Ausgang wird damit zu einem Analogausgang. Mit diesem Programm erhalten Sie eine stufenweise von 0 V bis 4,5 V ansteigende Gleichspannung. Verfolgen Sie den Spannungsverlauf mit einem Messgerät oder einem Oszilloskop.

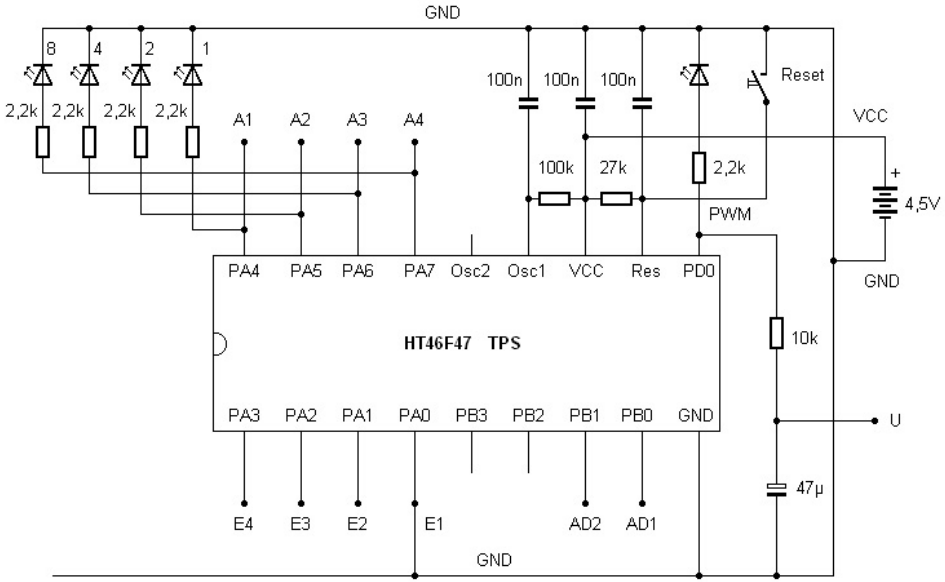

Abb. 7: Tiefpassfilter am PWM-Ausgang

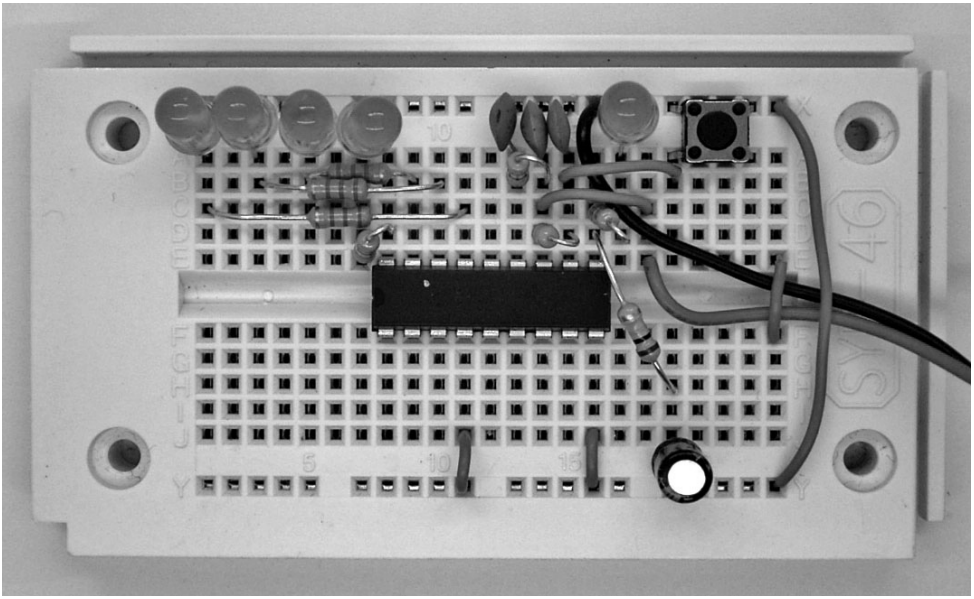

Abb. 8: Geglättete PWM-Ausgangsspannung

4 Analog-Digital-Wandler

Mit einer Verbindung E2 gegen GND und einem Druck auf die Reset-Taste starten Sie ein kleines Beispielprogramm zum Analog-Digital-Wandler (AD-Wandler). Die analoge Spannung am analogen Eingang AD1 wird gemessen und in einen digitalen Zahlenwert umgesetzt. Weil der TPS-Controller durchgehend mit 4-Bit-Werten arbeitet, ist das Ergebnis der Analog-Digital-Wandlung eine Zahl im Bereich 0 bis 15. Das Ergebnis 0 steht für die Eingangsspannung 0, das Ergebnis 15 für eine Spannung, die der Betriebsspannung entspricht, also z. B. 4,5 V. Der AD-Wert wird als Binärzahl an den vier LEDs ausgegeben und zusätzlich an den PWM-Ausgang übergeben. Schließen Sie einen Spannungsteiler aus einem Festwiderstand und einem lichtabhängigen Sensorwiderstand (LDR) an den analogen Eingang AD1.

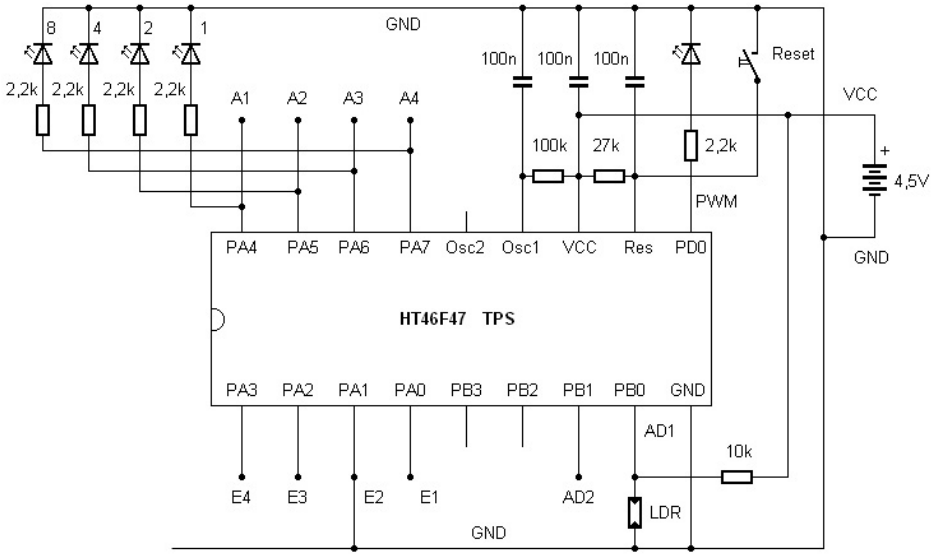

Abb. 9: Anschluss des Lichtsensors

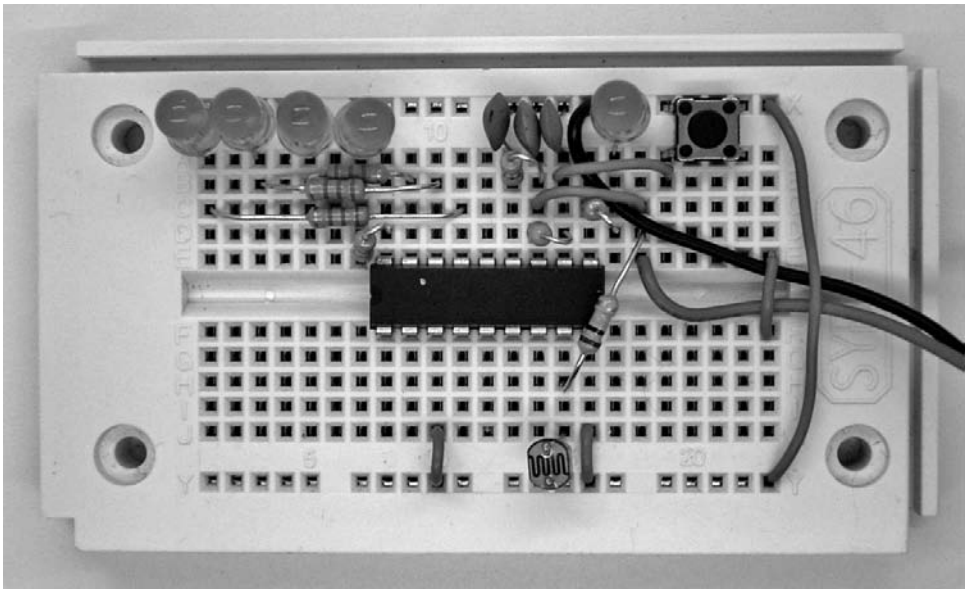

Abb. 10: Der LDR am AD1-Eingang

Das Beispielprogramm hat wegen der Ausgabe an die digitalen Ausgänge und den PWM-Ausgang große Ähnlichkeit mit dem Programm aus dem letzten Abschnitt. In der ersten Zeile steht jedoch das Kommando zum Wandeln eines Analogwerts.

Adresse	Befehl	Daten	Kommentar
2A	6	9	A = AD1
2B	5	4	Port = A
2C	5	9	PWM = A
2D	2	6	Warte 100 ms
2E	3	4	Springe -4

Listing 3: AD-Wandler und PWM-Ausgang

Testen Sie das Programm mit unterschiedlicher Beleuchtung des Sensors. Je mehr Licht auf den LDR fällt, desto kleiner ist die Spannung an AD1. Umgekehrt entstehen bei Dunkelheit maximale AD-Werte und damit eine maximale Helligkeit der LED am PWM-Ausgang. Lesen Sie die Binärzahlen vom LED-Display ab und versuchen Sie z. B. eine Helligkeit genau auf der Hälfte des Bereichs einzustellen. Der digitale Wert liegt dann bei 0111 oder 1000. Bei etwas flackerndem Kunstlicht kann es vorkommen, dass das Ergebnis zwischen zwei Stufen wechselt.

5 Zufallsgenerator

Mit einer Drahtbrücke E3 gegen GND starten Sie ein Beispielprogramm für einen Zufallsgenerator. Hier wird die Taste S1 ausgewertet. Der zugehörige Eingang verfügt über einen internen Pullup-Widerstand, der die Spannung auf VCC-Level zieht. Die Taste ist gegen Masse angeschlossen. Ein Tastendruck zieht den Eingang S1 auf Null.

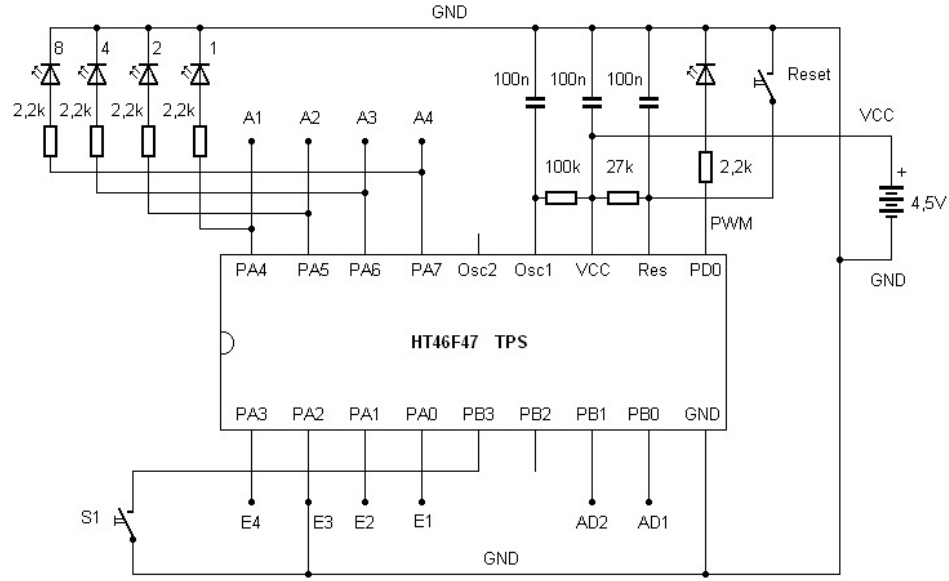

Abb. 11: Start des Zufallsschalters

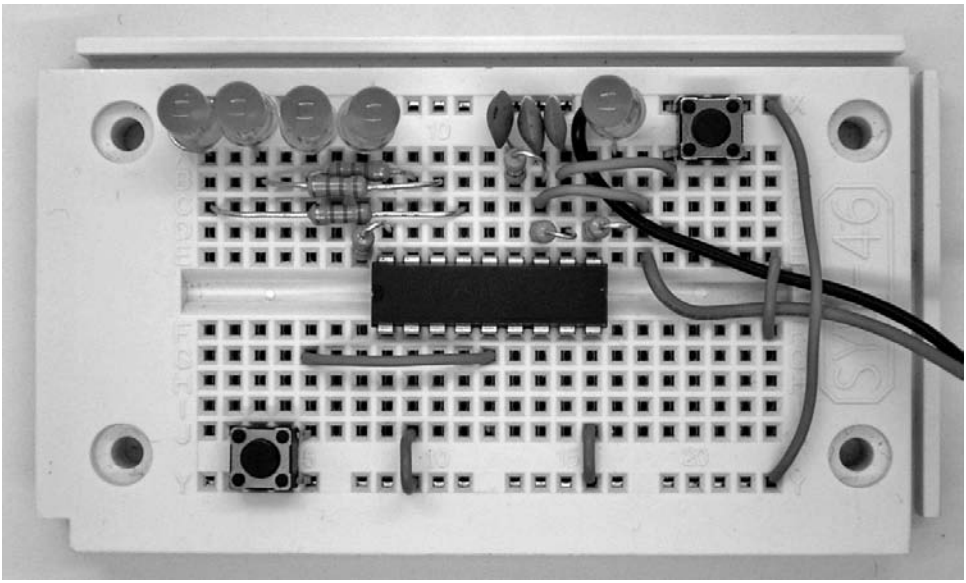

Abb. 12: Drahtbrücke zwischen E3 und GND

Das Programm verwendet einen bedingten Sprungbefehl. Wenn der S1-Eingangszustand Eins ist, wird der folgende Befehl übersprungen. Drückt man auf die Taste, ist der Zustand Null, und damit wird die Erhöhung der Variablen A ausgeführt. Das führt zu einem schnellen Hochzählen des Ausgangszustands. Beim Loslassen bleibt der letzte Zählerstand stehen. Wegen der hohen Zählgeschwindigkeit hat man keinen Einfluss auf das Ergebnis, das also zufällig ist.

Adresse	Befehl	Daten	Kommentar
30	5	4	Port = A
31	C	E	S1 = 1?
32	7	1	A = A + 1
33	3	3	Springe -3

Listing 4: Zufallsgenerator

Drücken Sie jeweils kurz auf die Taste, um ein neues Zufallsergebnis zu erhalten. Testen Sie die Zufallsfunktion, indem Sie eine Statistik der Ergebnisse erstellen. Nach einer ausreichend großen Zahl von Durchläufen sollte sich zeigen, dass alle Ergebnisse ungefähr gleich häufig vorkommen. Das Programm eignet sich auch als Spiel, bei dem z. B. die Zahl 1111 »gewürfelt« werden muss.

Zugleich ist das Programm ein Zähler mit der maximal möglichen Arbeitsgeschwindigkeit, weil kein Wartebefehl verwendet wird. Sie können daher an diesem Beispiel die Arbeitsgeschwindigkeit des TPS-Controllers untersuchen. Solange die Taste gedrückt wird, erscheint am Ausgang A1 ein Rechtecksignal mit einer Frequenz von ca. 133 Hz und einer Periodendauer vom 7,5 ms. Der Port ändert also nach jeweils 3,75 ms seinen Zustand. Das Programm durchläuft in der Zählschleife vier Befehle. Pro Befehl wird also etwa eine Millisekunde gebraucht. Der letzte Ausgang A4 zeigt eine Frequenz von 16,6 Hz, was noch als sichtbares Flackern erkennbar ist.

Wenn für zeitkritische Aufgaben einmal höhere Arbeitsgeschwindigkeiten erforderlich sind, können Sie die Taktgeschwindigkeit des Controllers durch Verkleinern des Widerstands an Osc1 erhöhen. Mit 100 kΩ hat man eine Taktrate von 2 MHz. Tauschen Sie den Widerstand gegen einen mit 27 kΩ. Damit erreichen Sie eine fast vierfach höhere Taktrate und eine Befehlszeit von ca. 0,25 ms. Im Normalfall sollte der Controller jedoch mit 100 kΩ an Osc1 laufen. Damit ist eine geringe Stromaufnahme und sicheres Arbeiten auch bei kleiner Betriebsspannung bis herunter auf 2,2 V gesichert.

6 Impulslängenmessung

Mit E4 an GND wird nach einem Reset ein Beispielprogramm zur Messung einer Impulslänge gestartet. Auch hier wird wieder der Zustand am Eingang S1 ausgewertet.

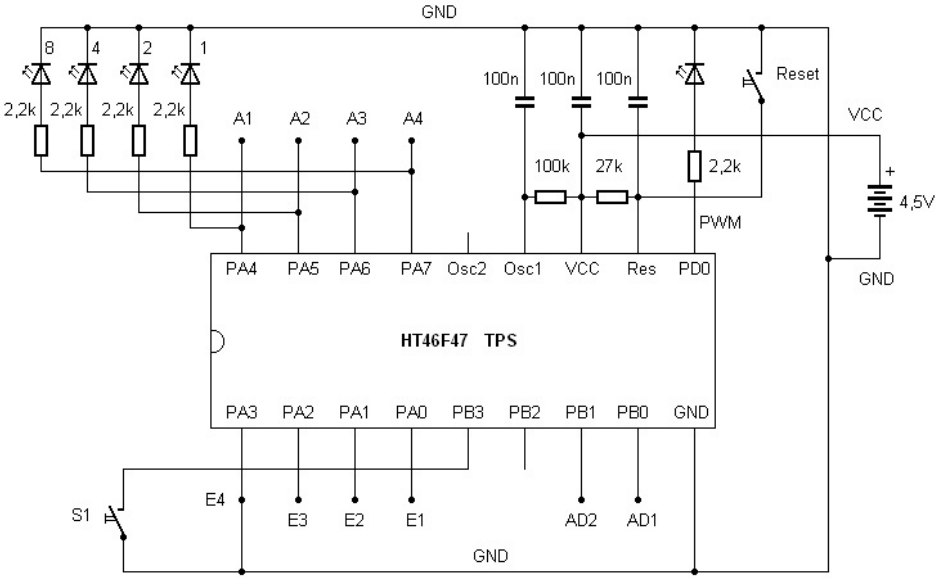

Abb. 13: E4 an GND

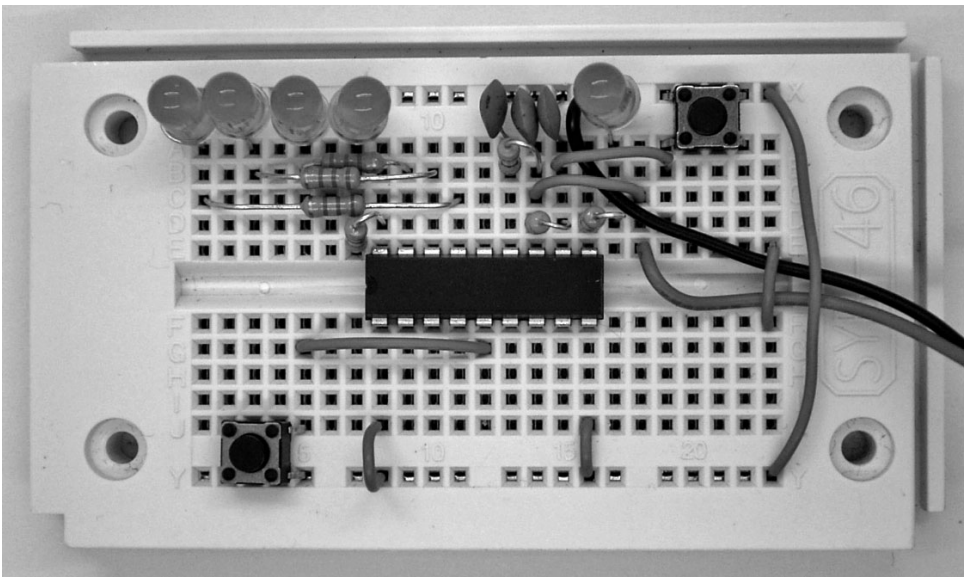

Abb. 14: Start der Impulslängenmessung

Eine Zeitmessung läuft im Zustand S1 = 0, also bei gedrückter Taste. Zur Wartezeit von 5 ms kommen noch einmal ca. 5 ms für die Ausführung von insgesamt fünf Befehlen in der Zählschleife. Die Zeiteinheit der Messung ist daher 10 ms.

Adresse	Befehl	Daten	Kommentar
34	2	2	Warte 5 ms
35	C	C	S1 = 0?
36	3	2	Springe -2
37	4	0	A = 0
38	2	2	Warte 5 ms
39	7	1	A = A + 1
3A	5	4	Port = A
3B	C	E	S1 = 1?
3C	3	4	Springe -4
3D	3	9	Springe -9

Listing 5: Zeitmessung

Drücken Sie die Taste S1 möglichst kurz. Sie erhalten z. B. das Ergebnis 1010, also dezimal 10. Da die Zeiteinheit des Programms 10 ms beträgt, bedeutet die Anzeige 100 ms. Mit etwas Training erreicht man auch kürzere Zeiten bis herunter auf 50 ms.

7 Programme auslesen

Für die Programmierung werden die Tasten S1 (Dateneingabe, links) und S2 (Programmieren, rechts) gebraucht. Außerdem ist die Reset-Taste erforderlich. Allein mit diesen Tasten können Programme ausgelesen und beliebige Programme eingegeben werden. Mit etwas Übung kann man in kürzester Zeit neue Programme eingeben und bestehende Programme modifizieren.

In den Programmiermodus gelangt man mit einem Reset bei gedrückter Programmiertaste S2, wobei S2 erst etwa eine halbe Sekunde nach Reset losgelassen werden darf. Man kann nun allein mit der Taste S2 durch das vorhandene Programm scrollen und die Befehle und Daten ansehen. Jede Adresse erfordert dazu zwei Tastenbetätigungen an S2. So wechselt man zwischen Anzeige des Befehls und der Daten. Außerdem wird jeweils für kurze Zeit die aktuelle Adresse angezeigt.

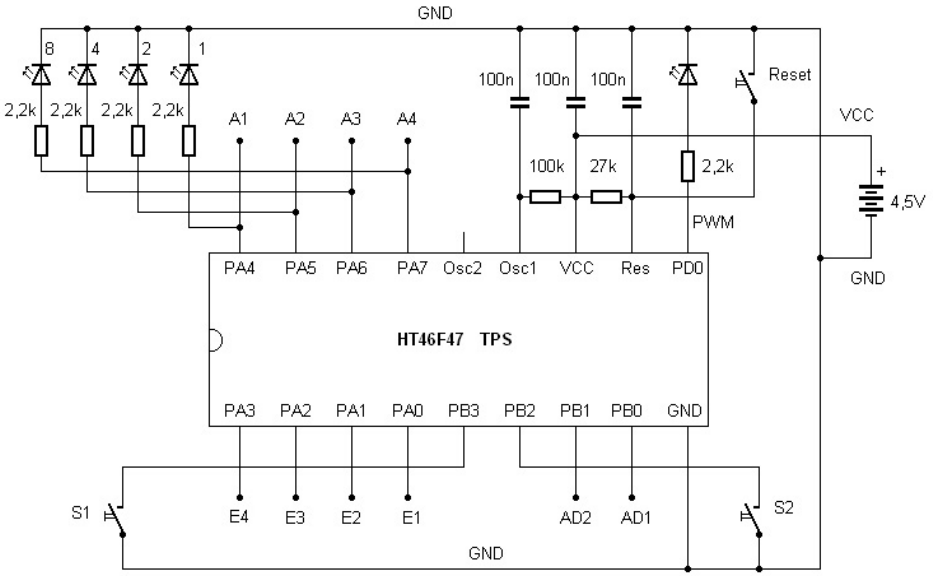

Abb. 15: S1 und S2 für den Programmiermodus

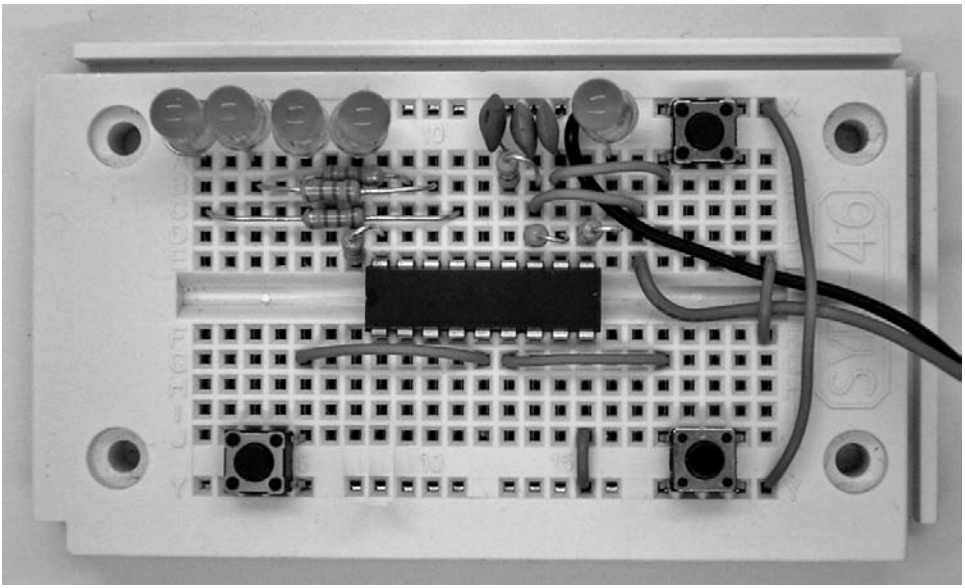

Abb. 16: Drei Taster und LED-Anzeige

- Erster Tastendruck S2
- Adresse (untere vier Bit) anzeigen, 300 ms
- Anzeige aus, 300 ms
- Befehl anzeigen
- Zweiter Tastendruck S2
- Daten anzeigen
- Dritter Tastendruck S2
- Nächste Adresse anzeigen, 300 ms
- usw.

Will man z. B. ein bestehendes Programm mit fünf Schritten nur ansehen, aber nicht verändern, gelangt man mit insgesamt zehn Betätigungen von S2 bis ans Ende. Weil jeweils die aktuelle Adresse kurz eingeblendet wird, fällt die Orientierung leicht. Man weiß immer, ob die Anzeige gerade einen Befehl oder Daten darstellt. Im Auslieferungszustand befinden sich die folgenden Befehle in den ersten fünf Adressen. Dabei handelt es sich um den Anfang des Auswahlprogramms zum Start der einzelnen Beispielprogramme.

Adresse	Befehl	Daten	Kommentar
00	6	4	A = Din
01	5	1	B = A
02	4	E	A = 14
03	8	0	AdrHi = 0
04	C	3	A = B?

Listing 6: Programmcode im Grundzustand

Ein 4-Bit-Befehl und die zugehörigen 4-Bit-Daten bilden zusammen ein Byte, also eine 8-Bit-Zahl. Ein halbes Byte bezeichnet man auch als »Nibble«. Das höherwertige Nibble bildet jeweils den Befehl, das niederwertige Nibble die zugehörigen Daten. Das EEPROM des Controllers fasst insgesamt 128 Bytes. Damit kann ein Programm maximal 128 Befehle umfassen. Das reicht für die meisten Anwendungen aus, weil der Programmcode extrem kompakt ist. Viele nützliche Programme kommen mit weniger als zehn Befehlen aus.

Bringen Sie die einzelnen Befehle und Daten in die Anzeige und vergleichen Sie den Inhalt des Speichers. Drücken Sie danach erneut auf die Reset-Taste. Das alte Programm startet unverändert.

8 Programme eingeben

Die Taste S1 kommt nur zur Anwendung, wenn man einen Befehl oder seine Daten verändern oder neu eingeben will. Grundsätzlich können nur Zahlenwerte zwischen 0 und 15 eingegeben werden. Mit dem ersten Druck auf S1 wird 0 eingestellt. Jeder folgende Tastendruck erhöht die Zahl um 1. Der aktuelle Stand wird jeweils über die vier LEDs binär angezeigt. Will man z. B. 4 eingeben, drückt man insgesamt fünfmal auf S1: 0, 1, 2, 3, 4. Die binäre Anzeige lautet dann 0100.

Wenn auf diese Weise entweder der Befehl oder die Daten oder beides neu eingegeben wurde, führt der zweite Tastendruck auf S2 dazu, dass dieses Byte ins EEPROM programmiert wird. Um das zu verdeutlichen, wird die LED-Anzeige für 600 ms abgeschaltet, bevor die nächste Adresse und danach der nächste Befehl angezeigt wird. Diese kleine Pause soll intuitiv als Programmiervorgang verstanden werden. Man kann im

Hinterkopf die Vorstellung aufbauen, dass das System die Energie für die Anzeige einspart und für die Programmierung des EEPROM verwendet. So etwas kennt man auch vom Auto: Wenn der Anlasser betätigt wird, gehen für einen kurzen Moment Licht und Radio aus.

Man kann ein bereits bestehendes Programm auch nur an einer Stelle verändern. Mit S2 scrollt man dann bis zur gewünschten Stelle und verändert mit S1 den Befehl oder die Daten, um sie dann mit S2 zu speichern.

Für den ersten Test wird ein Programm mit nur zwei Befehlen eingegeben. Es schaltet drei LEDs ein und führt dann in eine Endlosschleife.

Adresse	Befehl	Daten	Kommentar
00	1	7	A1-4 = 0111
01	3	0	Springe 0

Listing 7: LEDs einschalten

Statt eines ausführlichen Listings kann man auch eine Kurzschreibweise wählen. Dabei werden die beiden Bytes zu Hexzahlen zusammengefasst: 17h, 30h. Im Folgenden wird grundsätzlich die hexadezimale Schreibweise verwendet. Die Programme werden daher in kurzer Schreibweise ohne die Hex-Kennzeichen geschrieben: 17 30

Zur Eingabe muss Folgendes getippt werden:

S2 + Reset
2 x S1
S2
8 x S1
S2
4 x S1
S2
1 x S1
S2

Wenn man versehentlich einmal zu oft auf die Taste S1 getippt hat, kann die richtige Zahl trotzdem erreicht werden. Man muss dann noch einmal über 15 gehen, denn danach folgt ein Übertrag auf den Wert 0.

Nach der vollständigen Eingabe wird das neue Programm mit der Reset-Taste gestartet. Man sieht, dass drei LEDs angeschaltet werden. Weiter passiert nichts. Der Controller reagiert nun auch nicht mehr auf die Zustände an den Eingängen E1 bis E4, da das ursprüngliche Programm teilweise überschrieben wurde. Damit können auch die Beispielprogramme nicht mehr gestartet werden.

Da Sie nur die ersten beiden Speicheradressen geändert haben, können Sie das ursprüngliche Programm leicht wieder in Gang setzen. Hierzu geben Sie nur die ersten beiden Kommandos (64 51) entsprechend dem Listing aus dem letzten Abschnitt neu ein.

Testen Sie die ursprüngliche Funktion der Beispielprogramme. Geben Sie am besten das neue Übungsprogramm noch einmal ein. Nach kurzer Zeit gewinnen Sie Sicherheit im Umgang mit der Programmeingabe.

9 Wiederherstellung der Beispielprogramme

Falls Sie nach einiger Zeit den Urzustand des Controllers wiederherstellen möchten, kann dies mit der Eingabe von zwei Bytes FF geschehen. Tatsächlich entspricht dies dem Zustand des unbeschriebenen EEPROM. Die Firmware des TPS-Controllers enthält eine Startfunktion, die zunächst die ersten beiden Adressen überprüft, um einen leeren Speicher zu erkennen. Werden hier zwei FF-Bytes gelesen, geht der Controller davon aus, dass noch kein Programm eingegeben wurde. In diesem Fall wird das EEPROM automatisch mit der Beispiel-Software gefüllt. Diese Funktion dient eigentlich dazu, den Controller beim ersten Start mit den Beispielprogrammen im EEPROM zu versehen. Sie kann aber jederzeit verwendet werden, um den Grundzustand wiederherzustellen.

Adresse	Befehl	Daten	Kommentar
00	F	F	-
01	F	F	-

Listing 8: Rückkehr zum Grundzustand

Starten Sie also den Programmiermodus durch Reset bei gedrückter S2-Taste. Geben Sie dann insgesamt viermal den Wert F (dezimal 15) ein, bei dem alle LEDs A1 bis A4 an sind. Schließen Sie auch die letzte Eingabe mit S2 ab.

Drücken Sie dann auf Reset. Der Controller braucht nur einen Moment länger als üblich, um alle Bytes der Beispielprogramme neu einzuprogrammieren. Damit ist der Urzustand wiederhergestellt. Testen Sie z. B. ohne Drahtbrücke an den Eingängen den Wechselblinker von Seite 8.

10 TPS-Grundbefehle

Die tastenprogrammierbare Steuerung kennt insgesamt 14 Befehle (1-14). Zu vielen dieser Befehle gehört ein Parameter in Form einer 4-Bit-Zahl 0000 bis 1111 (0-F), also mit einem Zahlenbereich bis 15 (dezimal). Andere Befehle kennen Unterfunktionen, die in Form des Parameters angegeben werden. Hinter einem Befehlscode können sich daher bis zu 16 Unterbefehle verbergen. So steht z. B. der Befehl 7 für »Rechne A = ...«. Der Parameter gibt an, welche Rechenfunktion ausgeführt werden soll.

Im Folgenden werden Befehle und Daten zusammen in hexadezimaler Schreibweise als ein Byte geschrieben. Aus dem Befehl 1 zusammen mit dem Parameter 4 wird so das Kommando 14h. Das Hex-Kennzeichen wird weggelassen, weil alle Befehle und Adressen grundsätzlich in hexadezimaler Schreibweise stehen.

Die ersten drei Befehle lauten:

10-1F: direkte Port-Ausgabe an A1-A4, 0-15, binär 0000 bis 1111

20-2F: Wartezeit 0-15
(1, 2, 5, 10, 20, 50, 100, 200, 500, 1.000, 2.000, 5.000, 10.000, 20.000, 30.000, 60.000 ms)

30-3F: Sprung zurück 0-15

Befehl 1 dient zur Port-Ausgabe einer konstanten Zahl. Damit lassen sich beliebige Bitmuster ausgeben und z. B. auch mehrere LEDs gleichzeitig einschalten.

Der Wartebefehl 2 verwendet einen Parameter, der die Zeit in Millisekunden und in einer 1-2-5-Stufung enthält. Trotz des geringen Zahlenumfangs von 0 bis 15 lassen sich auf diese Weise Verzögerungszeiten zwischen einer Millisekunde und einer Minute ausführen. Noch längere Zeiten müsste man durch mehrmaliges Ausführen eines Wartebefehls, z. B. in einer Zählschleife, programmieren.

Der Rücksprungbefehl 3 ist besonders einfach und reicht für viele Aufgaben aus, bei denen ein Vorgang endlos wiederholt werden soll. Die Sprungweite ist auf den Bereich bis 15 begrenzt. Da die Sprungweite relativ zur aktuellen Adresse gilt, können Programmteile mit diesem Rücksprung beliebig an andere Adressen verschoben werden.

Das Wechselblinkerprogramm kommt mit diesen drei Befehlen aus. Es soll hier leicht modifiziert in den Adressbereich ab 00 geschrieben werden. Auch die Ausgabe-Bitmuster und die Wartezeiten wurden verändert.

Adresse	Befehl	Daten	Kommentar
00	1	1	A1-4 = 0001
01	2	7	Warte 200 ms
02	1	4	A1-4 = 0100
03	2	7	Warte 200 ms
04	3	4	Springe -4

Listing 9: Blinkprogramm

In hexadezimaler Kurzschreibweise sieht das Programm nun so aus:

11 27 14 27 34

Auf der Basis dieser ersten drei Befehle lassen sich bereits zahlreiche einfache Programme schreiben. Analysieren und testen Sie die folgenden drei Programme. Das Ziel sollte sein, dass Sie diese Befehle intuitiv anwenden können. Einfache Programmabläufe wie diese kann man nach einiger Übung sogar aus dem Kopf programmieren und direkt eingeben. Ein Beispiel dafür ist ein einfaches Lauflicht mit vier Ausgabemustern:

Adresse	Befehl	Daten	Kommentar
00	1	1	LEDs 0001
01	2	8	Warte 500 ms
02	1	2	LEDs 0010
03	2	8	Warte 500 ms
04	1	4	LEDs 0100
05	2	8	Warte 500 ms
06	1	8	LEDs 1000
07	2	8	Warte 500 ms

Adresse	Befehl	Daten	Kommentar
08	3	8	Springe -8

11 28 12 28 14 28 18 28 38

Listing 10: Lauflicht 1

Erweitern Sie das Programm um zwei weitere Ausgabemuster, sodass der Leuchtpunkt immer hin- und zurückläuft. Experimentieren Sie auch mit anderen Ausgabemustern und Verzögerungszeiten.

Adresse	Befehl	Daten	Kommentar
00	1	1	LEDs 0001
01	2	8	Warte 500 ms
02	1	2	LEDs 0010
03	2	8	Warte 500 ms
04	1	4	LEDs 0100
05	2	8	Warte 500 ms
06	1	8	LEDs 1000
07	2	8	Warte 500 ms
08	1	4	LEDs 0100
09	2	8	Warte 500 ms
0A	1	2	LEDs 0010
0B	2	8	Warte 500 ms
0C	3	C	Springe -12

11 28 12 28 14 28 18 28 14 28 12 28 3C

Listing 11: Lauflicht 2, hin und zurück

Ein Zeitschalter kann mit einem Wartebefehl eine Verzögerung von bis zu einer Minute enthalten. Am Ende steht ein Rücksprung mit der Sprungweite 0, also eine Endlosschleife ohne Inhalt, die als Programmende dient. Ein neuer Start wird mit dem Reset-Taster ausgelöst. Erweitern Sie das Programm auch einmal zu einem Drei-Minuten-Küchen-Timer. Dabei könnten Sie die verbleibende Zeit durch die Anzahl der leuchtenden LEDs als Lichtbalken darstellen.

Adresse	Befehl	Daten	Kommentar
00	1	F	LEDs 1111
01	2	F	Warte 1 min
02	1	0	LEDs 0000
03	3	0	Ende

1F 2F 10 30

Listing 12: Zeitschalter für eine Minute

11 Rechnen mit Variablen

Bisher wurden in den Parametern der einzelnen Befehle nur konstante Zahlenwerte verwendet. Das ist sinnvoll, wenn ein Programm immer gleich ablaufen soll. Komplexere Programme arbeiten dagegen mit variablen Daten. Es kann z. B. eine Rechnung wie A = A + B ausgeführt werden. Abhängig vom Inhalt der Variablen A und B wird dabei jedes Mal etwas anderes herauskommen. Das Ergebnis könnte z. B. die LEDs an den Ausgängen steuern.

Die Steuerung besitzt die vier Variablen A, B, C und D. Die wichtigste Variable ist A. Sie wird auch als *Akkumulator* oder kurz *Akku* bezeichnet. A ist bei allen Rechenoperationen beteiligt und erhält das Rechenergebnis. Außerdem wird A für den Datentransport eingesetzt. B wird hauptsächlich für Rechenoperationen benötigt. C und D können als Zwischenspeicher dienen und werden später noch als Zähler für Zählschleifen gebraucht.

Außerdem gibt es zwei analoge Eingänge (AD1 und AD2) und einen PWM-Ausgang. Die verarbeiteten Daten sind auf vier Bit begrenzt und nur über die Variable A zugänglich (A = AD1, PWM = A). Der Akku A kann auch direkt mit einer Zahl geladen werden (Befehle 40-4F). Um B, C oder D zu füllen, muss man zuerst A laden und den Inhalt dann der anderen Variablen zuweisen (Befehle 51-53). Mit A und B können einige Rechenschritte (Befehle 71-7A) durchgeführt werden.

Die Befehle 40-4F weisen A einen neuen Wert zu. Die Befehlsgruppe 51-5A überträgt den Inhalt von A an ein Ziel wie z. B. eine andere Variable oder den PWM-Ausgang. In dieser Gruppe gibt es auch Kommandos, die ein einzelnes Bit des Ausgangs-Ports setzen.

Genau die andere Datenrichtung liegt mit der Befehlsgruppe 61-6A vor, wo Daten einer Quelle in A eingelesen werden. Die Befehlsgruppe 71-7A schließlich führt einige Rechenoperationen durch, wobei das Ergebnis grundsätzlich in A erscheint. Der Ausgangs-Port Dout umfasst die vier Ausgänge A1 bis A4, die entweder gemeinsam oder als Einzel-Bits Dout.0 bis Dout.3 angesprochen werden können. Die Eingänge E1 bis E4 werden in gleicher Weise als Eingangs-Port Din angesprochen.

40-4F: A = 0-15

51-5A: Ziel 1-9 = A
51: B = A
52: C = A
53: D = A
54: Dout = A
55: Dout.0 = A.0
56: Dout.1 = A.0
57: Dout.2 = A.0
58: Dout.3 = A.0
59: PWM = A

61-6A: A = Quelle 1-10
61: A = B
62: A = C
63: A = D
64: A = Din

65: A = Din.0
66: A = Din.1
67: A = Din.2
68: A = Din.3
69: A = AD1
6A: A = AD2

71 -7A: A = Ausdruck 1-10
71: A = A + 1
72: A = A - 1
73: A = A + B
74: A = A - B
75: A = A * B
76: A = A / B
77: A = A And B
78: A = A Or B
79: A = A Xor B
7A: A = Not A

Ein Beispiel für die Verwendung der Variablen A findet man unter den Programmbeispielen in Kapitel 3. Das Programm wurde hier an die Adresse Null gesetzt und leicht erweitert. Hinzugekommen ist ein definierter Anfang mit dem Wert 0 in der Variablen A. In Adresse 01 findet man einen Rechenbefehl, hier eine Erhöhung um 1. Der Inhalt der Variablen A wird danach an den PWM-Ausgang und an den Ausgangs-Port übergeben.

Adresse	Befehl	Daten	Kommentar
00	4	0	A = 0
01	7	1	A = A + 1
02	5	4	Port = A
03	5	9	PWM = A
04	2	6	Warte 100 ms
05	3	4	Springe -4

40 71 54 59 26 34
Listing 13: Erhöhen um 1

Ein weiteres Beispiel wurde bereits in Kapitel 4 gezeigt. Die Daten kommen dabei vom Analogeingang AD1 und werden an den Ausgangs-Port und an den PWM-Ausgang übertragen. Das modifizierte Programm enthält noch einen zusätzlichen Rechenschritt, nämlich die Invertierung des Inhalts der Variablen A. Dadurch wird aus dem Wert 0000 der neue Wert 1111, d. h., aus 0 wird 15 und umgekehrt. Eine ansteigende Eingangsspannung führt auf diese Weise zu einer absteigenden PWM-Ausgabe.

Adresse	Befehl	Daten	Kommentar
00	6	9	A = AD1
01	5	4	Port = A
02	7	A	A = Not A
03	5	9	PWM = A
04	2	6	Warte 100 ms
05	3	5	Springe -5

69 54 7A 59 26 35

Listing 14: Invertieren

12 Sprünge und Verzweigungen

Bisher gab es nur einen einfachen Rücksprung (Befehl 3), der maximal 15 Adressen zurückreichte. Nun kommt ein absoluter Sprung (Jump) hinzu. Da das Sprungziel nur mit 4 Bit angegeben werden kann, gibt es einen zusätzlichen Befehl, der das High-Nibble der Adresse festlegt. Damit hat man den Adressraum 0-255. Das ist mehr, als benötigt wird, denn das EEPROM des Controllers fasst nur 128 Bytes, also den Bereich 00 bis 7F (dezimal 0 bis 127). Der Speicher ist damit praktisch in acht Seiten, Seite 0 bis Seite 7 (Page 0-7), eingeteilt. Vor einem absoluten Sprung muss die Seite des Sprungziels festgelegt werden.

Zwei Zählschleifen mit den Variablen C und D führen ebenfalls absolute Sprünge aus, wobei auch hier zuvor die Seite der Adresse festgelegt werden muss.

Die bedingten Sprünge arbeiten als Skip-Befehle (Auslassen, Überspringen). Wenn die jeweilige Bedingung wahr ist, wird eine Adresse übersprungen. Dort könnte z. B. ein Sprungbefehl oder auch ein Rechenbefehl stehen. Als Bedingungen stehen Vergleiche zwischen A und B sowie direkte Bit-Abfragen des Eingangs-Ports zur Verfügung.

Außerdem gibt es noch einen Unterprogrammaufruf (Call) und den dazugehörenden Rücksprungbefehl (Return). Es sind zwar mehrere Unterprogramme erlaubt, aber aus einem Unterprogramm darf kein weiteres Unterprogramm aufgerufen werden, weil der Interpreter sich immer nur eine Rücksprungadresse merkt.

80-8F: Adr-high = 0-15

90-0F: direkter Sprung (Jump) auf Adr-high, Adr-low (0-15)

A0-AF: Zählschleife C-mal Adr-high, Adr-low (0-15)

B0-BF: Zählschleife D-mal Adr-high, Adr-low (0-15)

C1-CF: bedingter Sprung: wenn (Bedingung 1-15) dann überspringe
C1: if A > B then Adr = Adr + 1
C2: if A < B then Adr = Adr + 1
C3: if A = B then Adr = Adr + 1
C4: if Din.0 = 1 then Adr = Adr + 1

C5: if Din.1 = 1 then Adr = Adr + 1
C6: if Din.2 = 1 then Adr = Adr + 1
C7: if Din.3 = 1 then Adr = Adr + 1
C8: if Din.0 = 0 then Adr = Adr + 1
C9: if Din.1 = 0 then Adr = Adr + 1
CA: if Din.2 = 0 then Adr = Adr + 1
CB: if Din.3 = 0 then Adr = Adr + 1
CC: if S1 = 0 then Adr = Adr + 1
CD: if S2 = 0 then Adr = Adr + 1
CE: if S1 = 1 then Adr = Adr + 1
CF: if S2 = 1 then Adr = Adr + 1

D0-DF: Unterprogrammaufruf (Call) Adr-high, Adr-low (0-15)

E0-EF: Rücksprung vom Unterprogramm (Return)

Ein Beispiel für die Verwendung bedingter Sprungbefehle findet man im Beispielprogramm in Kapitel 6. Es wurde hier leicht modifiziert an die Adresse 0 gesetzt. Weil der obere Teil der Adresse (Adr-hi) im Ruhezustand 0 ist, der Controller also auf der Seite 0 beginnt, muss der Befehl 80 hier nicht verwendet werden. Wieder wird die Länge eines Tastendrucks gemessen und angezeigt. Alle Wartebefehle wurden aus dem Programm entfernt, sodass es jetzt mit einer höheren Zeitauflösung arbeitet.

Adresse	Befehl	Daten	Kommentar
00	C	C	S1 = 0?
01	3	1	Springe -1
02	4	0	A = 0
03	7	1	A = A + 1
04	5	4	Port = A
05	C	E	S1 = 1?
06	3	3	Springe -3
07	3	7	Springe -7

CC 31 40 71 54 CE 33 37

Listing 15: Reaktion auf die Taste S1

Der Sprungbefehl CC in Adresse 00 wertet den Zustand am Taster S1 aus. Im Ruhezustand ist S1 = 1. Die Bedingung ist also nicht wahr und der Befehl in Adresse 01 wird nicht übersprungen. Dort steht ein relativer Sprungbefehl auf den Anfang. Das Programm wiederholt die Befehle in Adresse 00 und 01 so lange, bis die Taste gedrückt wird. Dann wird die Bedingung wahr und die Adresse 01 wird übersprungen. Damit beginnt der eigentliche Messvorgang. Der Akku wird gelöscht, dann immer wieder um 1 erhöht und an die LEDs ausgegeben. An der Adresse 05 steht ein weiterer bedingter Sprungbefehl CE. Hier ist die Bedingung für das Überspringen eines Befehls S1 = 1. Da die Taste zunächst noch gedrückt ist, ist die Bedingung nicht wahr. Der Befehl in 06 wird also ausgeführt und führt zu einem Rücksprung nach 03. Erst wenn die Taste losgelassen wird, gelangt das Programm an die Adresse 07 und damit zu einem Rücksprung auf den Anfang.

Geben Sie das Programm ein und testen Sie es. Die Reaktionszeit ist jetzt wesentlich schneller. Die Zeiteinheit liegt bei ca. 5 ms.

Das ursprüngliche Beispielprogramm befindet sich noch im Speicher ab Adresse 34h, da nur die unteren Adressen überschrieben wurden. Schreiben Sie ein kleines Programm, das nur einen Sprung auf diese Adresse enthält. Hier muss zuerst die Seite 3 angegeben werden. Der dann folgende absolute Sprung mit der angegebenen Adresse 4 zielt dann auf die tatsächliche Adresse 34.

Adresse	Befehl	Daten	Kommentar
00	8	3	Seite 3
01	9	4	Adresse = 34

83 94

Listing 16: Absoluter Sprung zum Zeitmesserprogramm

Das ursprüngliche Beispielprogramm wird damit wieder aufgerufen. Testen Sie dies auch einmal für andere Beispiele. Eine komplette Übersicht aller einsetzbaren Programme finden Sie im Anhang.

13 Befehlsübersicht

Alle Befehle auf einen Blick – das vereinfacht die Arbeit mit dem Controller. Die folgende Tabelle enthält den gesamten Befehlsvorrat in kompakter Form.

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

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

14 Zählschleifen

Ein Vorgang soll z. B. genau fünfmal ausgeführt werden. Dazu bildet man eine Zählschleife. Ein Sprungbefehl wird in diesem Fall fünfmal ausgeführt, danach nicht mehr. Die Zählvariable heißt C. Der Zählerwert 5 muss zuerst in A geladen werden und von da in C. Der Befehl A2 führt einen absoluten Sprung auf 02 aus und verkleinert zugleich den Inhalt der Variablen C um 1. Wenn C den Wert 0 erreicht hat, wird der Sprung nicht mehr ausgeführt. Die absolute Sprungadresse bezieht sich auf die angegebene Seite. Bei einem Programm auf der Seite 0 darf der Seitenbefehl 80 auch weggelassen werden. Er ist aber unbedingt nötig, wenn eine andere Seite angesprungen wird.

Adresse	Befehl	Daten	Kommentar
00	4	5	A = 5
01	5	2	C = A
02	1	5	Port = 0101
03	2	8	500 ms
04	1	A	Port = 1010
05	2	8	500 ms
06	8	0	Seite 0
07	A	2	C-mal 02
08	3	0	Ende

45 52 15 28 1A 28 80 A2 30

Listing 17: Eine Zählschleife

Testen Sie das Programm. Die LEDs zeigen bei jedem Durchgang die Muster 0101 und 1010. Allerdings wird dieser Programmteil offensichtlich nicht fünfmal, sondern genau sechsmal durchlaufen. Zwar wird der Sprungbefehl in Adresse 07 tatsächlich genau fünfmal ausgeführt, um aber zum ersten Mal an diese Stellen zu gelangen, wird schon ein Blinkvorgang durchgeführt. Deshalb blinkt das Programm insgesamt sechsmal. Ändern Sie die Zählvariable auf den Wert 4 und testen Sie das Programm erneut. Nun blinken die LEDs genau fünfmal.

Man kann die Zählschleife auch so verwenden, dass nicht zurück, sondern vorwärts gesprungen wird. Diesmal wird der Vorgang tatsächlich fünfmal ausgeführt, wenn C am Anfang mit dem Wert 5 geladen wurde. Die jeweils übersprungene Adresse 04 enthält einen relativen Sprung auf sich selbst, und damit eine Endlosschleife, die als Programmende dient.

Adresse	Befehl	Daten	Kommentar
00	4	5	A = 5
01	5	2	C = A
02	8	0	AdrHi = 0
03	A	5	C-mal 05
04	3	0	Ende
05	1	5	Port = 0101
06	2	8	Warte 500 ms
07	1	A	Port = 1010
08	2	8	Warte 500 ms
09	3	6	Springe -6

45 52 80 A5 30 15 28 1A 28 36

Listing 18: Fünfmal Blinken

15 Vergleiche

Zwei Zahlenwerte sollen verglichen werden. In Abhängigkeit vom Ergebnis des Vergleichs wird ein Sprung ausgeführt. Die beiden Zahlenwerte müssen sich in A und B befinden. Im folgenden Beispiel wird B mit der Zahl 5 geladen. A erhält sein Ergebnis vom Analogeingang AD1. Hier kann z. B., wie in Kapitel 4, ein Lichtsensor angeschlossen sein. Das Programm soll nun laufend die folgende Aktion durchführen:

Wenn AD1 > 5
dann: alle LEDs an
sonst: alle LEDs aus

Im Endergebnis erhält man einen Dämmerungsschalter. Da der LDR gegen GND angeschlossen ist, führt mehr Helligkeit zu einer kleineren Spannung an AD1. Die LEDs gehen aus, sobald eine gewisse Helligkeit überschritten und damit eine gewisse Spannung unterschritten wird. Der Grenzwert liegt bei 6, denn das Messergebnis muss größer als 5 sein.

Adresse	Befehl	Daten	Kommentar
00	4	5	A = 5
01	5	1	B = A
02	8	0	AdrHi = 0

Adresse	Befehl	Daten	Kommentar
03	6	9	A = AD1
04	C	1	Skip if A>B
05	9	8	Adr 08
06	1	F	LEDs 1111
07	3	4	Adr 03
08	1	0	LEDs 0000
09	3	6	Adr 03

45 51 80 69 C1 98 1F 34 10 36

Listing 19: Einfacher Dämmerungsschalter

Testen Sie das Programm, indem Sie den Lichtsensor mit der Hand mehr oder weniger abschatten. Sie werden feststellen, dass die Grundfunktion erfüllt ist. Allerdings kommt es meist zu einem unschönen Nebeneffekt. Genau an der Grenze zwischen An und Aus blinken die LEDs unkontrolliert. Vor allem bei Kunstlicht schwankt die Helligkeit in schneller Folge um einen gewissen Mittelwert. Dieses Flackern wird zwar vom Programm korrekt ausgewertet, das Ergebnis ist aber nicht so, wie man es von einem Dämmerungsschalter erwartet. Einen verbesserten Dämmerungsschalter zeigt das Kapitel 18.

16 AND, OR und XOR

Zwei binäre Zustände lassen sich zu einem neuen Zustand verknüpfen. Ein Beispiel ist die UND-Funktion: Wenn Bit 1 den Zustand 1 hat UND Bit 2 den Zustand 1, wird der Ausgangszustand ebenfalls 1. Auch Binärzahlen mit mehren Bits lassen sich auf diese Weise verknüpfen. Die Verknüpfung »10 AND 3 = 2« wird verständlich, wenn man sie in Binärzahlen schreibt:

1010 AND
0011 =
0010

Das folgende Programm verknüpft die Eingangszustände mit der konstanten Zahl 3. Die AND-Funktion bewirkt dabei praktisch, dass die beiden unteren Bits maskiert (herausgefiltert) werden. Im Ruhezustand hat der Eingangs-Port den Zustand 1111. Die UND-Verknüpfung mit 0011 liefert dann den Zustand 0011 an den LEDs. Legt man jedoch einen der Eingänge E1 oder E2 an GND, wird der 0-Zustand auch an den Ausgängen sichtbar. Änderungen an E3 und E4 haben keine Auswirkungen.

Adresse	Befehl	Daten	Kommentar
00	6	4	A = Din
01	5	1	B = A
02	4	3	A = 3

Adresse	Befehl	Daten	Kommentar
03	7	7	A = A And B
04	5	4	Port = A
05	3	5	Springe -5

64 51 43 77 54 35

Listing 20: Anwendung der AND-Funktion

Ändern Sie das Programm, und testen Sie auch andere logische Funktionen. Die OR-Funktion (78) kann verwendet werden, um bestimmte Eingangszustände grundsätzlich auf 1 zu setzen: 64 51 43 78 54 35

1010 OR
0011 =
1011

Mit der XOR-Funktion (Exklusiv-Oder, 79) kann man einzelne Bits invertieren: 64 51 43 79 54 35

1010 XOR
0011 =
1001

17 Unterprogramme

Wenn Teile eines Programms mehrfach verwendet werden sollen, schreibt man sie in ein Unterprogramm. Damit spart man oft Speicherplatz, manchmal aber auch viel Tipparbeit. Das folgende Beispiel demonstriert die Verwendung eines Unterprogramms, das an zwei Stellen im Hauptprogramm aufgerufen wird. Das Unterprogramm enthält in diesem Fall nur eine Anweisung (A = A-1) und den Rücksprungbefehl. Deshalb spart man hier keinen Speicherplatz, sondern das Beispiel dient nur zur Demonstration des CALL- und des RET-Befehls.

Hauptprogramm:

Adresse	Befehl	Daten	Kommentar
00	8	0	AdrHi = 0
01	D	8	Call 08
02	5	4	Ausgabe
03	2	9	Warte 1 s
04	D	8	Call 08
05	5	4	Ausgabe
06	2	8	Warte 0,5 s
07	3	7	Springe -7

Unterprogramm:

Adresse	Befehl	Daten	Kommentar
08	7	2	A = A-1
09	E	0	Ret

80 D8 54 29 D8 54 28 37 72 E0

Listing 21: Unterprogrammaufrufe

Das Ergebnis des Programms ist ein absteigender Binärzähler mit ungleichen Zeitverzögerungen. Testen Sie auch einmal andere Befehle im Unterprogramm.

Unter den im Auslieferungszustand vorhandenen Beispielprogrammen gibt es mehrere nützliche Unterprogramme für die allgemeine Verwendung. Sie sind im Anhang komplett aufgelistet. Für den eigenen Einsatz muss nur die Einsprungadresse bekannt sein:

50: Unterprogramm: Ton lang
52: Unterprogramm: Ton kurz
53: Unterprogramm: Ton beliebig, Länge in A
60: Unterprogramm: Warte auf Tastendruck an S1
68: Unterprogramm: Warte auf Tastendruck an S2
70: Unterprogramm: Zahleneingabe mit S1 und S2

Das Unterprogramm ab der Adresse 60 wird nun verwendet, um einen Zähler aufzubauen, der über die Taste S1 gesteuert wird. Der Zählerstand beginnt mit 0. Das Hauptprogramm ist relativ kurz, weil die komplexe Aufgabe der Tastenabfrage in das Unterprogramm ausgelagert wurde.

Adresse	Befehl	Daten	Kommentar
00	4	0	A = 0
01	5	4	Ausgabe
02	7	1	A = A+1
03	8	6	Seite 6
04	D	0	Call 60, Taste S1
05	3	4	Springe -4

40 54 71 86 D0 34

Listing 22: Über S1 gesteuerter Zähler

Testen Sie das Programm. Wenn Sie zehnmal auf S1 drücken, sollte das Ergebnis 1010 sein. Ändern Sie das Programm so ab, dass das Unterprogramm ab Adresse 68 verwendet wird. Nun reagiert der Zähler auf S2.

18 Dämmerungsschalter

Ein Dämmerungsschalter soll das Licht einschalten, wenn die Umgebungshelligkeit unter einen bestimmten Grenzwert fällt. Wenn es heller wird, soll umgekehrt das Licht wieder ausschalten. Es sollte sichergestellt werden, dass das Licht auf der Grenze zwischen hell und dunkel nicht flackert. Das gelingt mit einer Hysterese, also einem gewissen Abstand der Einschalt- und Ausschalthelligkeit. Das hier vorgestellte Programm arbeitet nach den folgenden Regeln:

- Wenn die Spannung an AD1 nicht größer als 5 ist, wird ausgeschaltet.
- Wenn die Spannung an AD1 nicht kleiner als 9 ist, wird eingeschaltet.

Damit hat man einen mittleren Bereich, in dem keine Änderung des Ausgangszustands eintreten kann. Diese Lücke verhindert ein Flackern der LEDs.

0-5: LEDs aus
6-8: LEDs unverändert
9-15: LEDs an

Adresse	Befehl	Daten	Kommentar
00	1	0	LEDs 0000
01	4	5	A = 5
02	5	1	B = A
03	6	9	A = AD1
04	C	1	Skip if A>B
05	1	0	LEDs 0000
06	4	9	A = 9
07	5	1	B = A
08	6	9	A = AD1
09	C	2	Skip if A<B
0A	1	F	LEDs 1111
0B	3	A	Springe -10

10 45 51 69 C1 10 49 51 69 C2 1F 3A

Listing 23: Dämmerungsschalter mit Hysterese

19 LED-Dimmer

Ziel dieses Programmbeispiels ist eine steuerbare LED-Lampe. Die Helligkeit einer LED am PWM-Ausgang soll über Tasten einstellbar sein. Man kann dabei jeweils kurz auf eine Taste drücken, um die nächste Helligkeitsstufe zu erreichen, oder man übt Dauerdruck aus, wobei die Helligkeit sich kontinuierlich verändert.

Im Kern des Programms kommen die schon bekannten Skip-Befehle zum Einsatz. Wenn die jeweilige Taste nicht gedrückt ist, wird der zugehörige Befehl zum Erhöhen oder Verkleinern des Akkuinhalts übersprungen. Die Schwierigkeit besteht aber darin, dass normalerweise dabei auch ein Überlauf von 15 auf 0 oder von 0 auf 15 auftreten kann. Es erfordert etwas mehr Aufwand, dieses Überlaufen zu verhindern. Dazu muss nämlich jeweils abgefragt werden, ob das untere Ende (0) oder das obere Ende (15) bereits erreicht ist. Da bei einem Vergleich grundsätzlich der Akku beteiligt ist, muss dessen Inhalt jeweils zwischengespeichert werden. Dazu wird die Variable C eingesetzt.

Adresse	Befehl	Daten	Kommentar
00	8	0	AdrHi = 0
01	5	9	PWM = A
02	2	7	200 ms
03	5	2	C = A
04	4	F	A = 15
05	5	1	B = A
06	6	2	A = C
07	C	2	Skip if A<B
08	9	B	Springe 0B
09	C	F	Skip if S2 = 1
0A	7	1	A = A + 1
0B	5	2	C = A
0C	4	0	A = 0
0D	5	1	B = A
0E	6	2	A = C
0F	C	1	Skip if A>B
10	9	0	Springe 00
11	C	E	Skip if S1 = 1
12	7	2	A = A - 1
15	9	0	Springe 00

80 59 27 52 4F 51 62 C2 9B CF 71 52 40 51 62 C1 90 CE 72 90

Listing 24: Helligkeitssteuerung

20 Zahlenschloss

Das hier vorgestellte Zahlenschloss schaltet den PWM-Ausgang ein, wenn der Benutzer die korrekte Zahlenfolge eingegeben hat. Die Zahleneingabe soll exakt nach dem Muster des Programmierens über die Tasten S1 und S2 ablaufen. Das folgende Programm demonstriert die Eingabe einer einzelnen Zahl über die Taste S1. Wie beim Programmiervorgang führt der erste Tastendruck zum Ergebnis 0000. Jeder folgende Druck auf S1 erhöht die Ausgabe um 1. Durch einen Druck auf S2 wird die Eingabe beendet. In diesem Fall endet das Programm in einer Endlosschleife.

Adresse	Befehl	Daten	Kommentar
00	C	C	S1 = 0?
01	3	1	Springe -1
02	4	0	A = 0
03	5	4	Dout = A
04	2	3	10 ms
05	C	E	S1 = 1?
06	3	2	Adr 04
07	C	F	S2 = 1?
08	3	0	Ende
09	C	C	S1 = 0?
0A	3	3	Adr 07
0B	7	1	A = A + 1
0C	2	3	10 ms
0D	C	C	S1 = 1?
0E	3	1	Adr 0D
0F	3	C	Adr 03

```
CC 31 40 54 23 CE 32 CF 30 CC 33 71 23 CC 31 3C
```

Listing 25: Eingabe einer Zahl

Die Zahleneingabe steht auch als fertiges Unterprogramm ab Adresse 70 zur Verfügung. Statt der Endlosschleife in Zeile 08 gibt es hier einen RET-Befehl. Das Unterprogramm wird mit dem Ergebnis der Zahleneingabe in A verlassen.

Das folgende Zahlenschloss ruft die Zahleneingabe dreimal auf und vergleicht die Ergebnisse mit vordefinierten Zahlen. In diesem Beispiel lautet die korrekte Eingabe 3, 5, 2. Danach wird der PWM-Ausgang mit dem Wert 15 voll aufgesteuert. Jede Fehleingabe führt dagegen in eine Endlosschleife, die nur durch einen Reset wieder verlassen werden kann.

Der PWM-Ausgang wird in diesem Beispiel wie ein normaler Digital-Port behandelt. Das ist erforderlich, weil alle vier Ausgänge A1 bis A4 für die Zahleneingabe benötigt werden. Nach jeder vollständigen Eingabe werden die vier LEDs gelöscht, um einem eventuellen Beobachter möglichst wenig Hinweise auf die geheime Zahlenkombination zu geben.

Adresse	Befehl	Daten	Kommentar
00	8	7	Page 7
01	4	3	A = 3
02	5	1	B = A
03	D	0	call 70
04	C	3	Skip if A=B
05	3	0	Ende
06	1	0	LEDs aus
07	4	5	A = 5
08	5	1	B = A
09	D	0	call 70
0A	C	3	Skip if A=B
0B	3	0	Ende
0C	1	0	LEDs aus
0D	4	2	A = 2
0E	5	1	B = A
0F	D	0	call 70
10	C	3	Skip if A=B
11	3	0	Ende
12	1	0	LEDs aus
13	4	F	A = 15
14	5	9	PWM=A
15	3	0	Ende

87 43 51 D0 C3 30 10 45 51 D0 C3 30 10 42 51 D0 C3 30 10 4F 59 30

Listing 26: Das Zahlenschloss

21 Anhang

Listing der Beispielprogramme

Adresse	Befehl	Daten	Kommentar
00	6	4	A = Din
01	5	1	B = A
02	4	E	A = 1110
03	8	0	Seite 0
04	C	3	A = B?
05	9	8	Springe 08
06	8	2	Seite 2
07	9	5	Springe 25, »Hochzählen«
08	4	D	A = 1101
09	8	0	Seite 0
0A	C	3	A =B ?
0B	9	E	Springe 0E
0C	8	2	Seite 2
0D	9	A	Springe 2A, »AD/PWM«
0E	4	B	A = 1011
0F	8	1	Seite 1

64 51 4E 80 C3 98 82 95 4D 80 C3 9E 82 9A 4B 81

Seite 0: Auswahl und Start der Beispielprogramme

Adresse	Befehl	Daten	Kommentar
10	C	3	A = B?
11	9	4	Springe 14
12	8	3	Seite 3
13	9	0	Springe 30, »Zufall«
14	4	7	A = 0111
15	8	1	Seite 1
16	C	3	A = B?
17	9	A	Springe 1A
18	8	3	Seite 3

Adresse	Befehl	Daten	Kommentar
19	9	4	Springe 34, »Stoppuhr S1«
1A	4	3	A = 0011
1B	8	2	Seite 2
1C	C	3	A = B?
1D	9	0	Springe 20 »Wechselblinker«
1E	8	4	Seite 4
1F	9	0	Springe 40 »Stoppuhr S1/S2«

C3 94 83 90 47 81 C3 9A 83 94 43 82 C3 90 84 90

Seite 1: Auswahl und Start der Beispielprogramme

Adresse	Befehl	Daten	Kommentar
20	1	1	0001 »Wechselblinker«
21	2	8	Warte 500 ms
22	1	8	1000
23	2	8	Warte 500 ms
24	3	4	Springe -4
25	7	1	A = A + 1 »Hochzählen«
26	5	4	Port = A
27	5	9	PWM = A
28	2	6	Warte 100 ms
29	3	4	Springe -4
2A	6	9	A = AD1 »AD/PWM«
2B	5	4	Port = A
2C	5	9	PWM = A
2D	2	6	Warte 100 ms
2E	3	4	Springe -4
2F	F	F	-

11 28 18 28 34 71 54 59 26 34 69 54 59 26 34 FF

Seite 2: Beispielprogramme: Wechselblinker, Hochzählen, AD/PWM

Adresse	Befehl	Daten	Kommentar
30	5	4	Port = A »Zufall«
31	C	E	S1 = 1?
32	7	1	A = A + 1
33	3	3	Springe -3
34	2	2	Warte 5 ms »Stoppuhr S1«
35	C	C	S1 = 0?
36	3	2	Springe - 2
37	4	0	A = 0
38	2	2	Warte 5 ms
39	7	1	A = A + 1
3A	5	4	Port = A
2B	C	E	S1 = 1?
3C	3	4	Springe -4
3D	3	9	Springe -9
3E	F	F	-
3F	F	F	-

54 CE 71 33 22 CC 32 40 22 71 54 CE 34 39 FF FF

Seite 3: Beispielprogramme: Zufall, Stoppuhr S1

Adresse	Befehl	Daten	Kommentar
40	8	6	Seite 6 »Stoppuhr Start/Stop«
41	D	0	Aufruf »Warte S1«
42	4	0	A = 0
43	7	1	A = A + 1
44	5	4	Port = A
45	2	3	Warte 10 ms
46	C	D	S2 = 0?
47	3	4	Springe -4
48	D	8	Aufruf »Warte S2«
49	4	0	A = 0
4A	5	4	Port = A
4B	3	B	Springe -11

Adresse	Befehl	Daten	Kommentar
4C	F	F	-
4D	F	F	-
4E	F	F	-
4F	F	F	-

86 D0 40 71 54 23 CD 34 D8 40 54 3B FF FF FF FF

Seite 4: Beispielprogramm Stoppuhr Start/Stopp

Adresse	Befehl	Daten	Kommentar
50	4	F	A = 15 »Ton lang«
51	9	3	Adr 03
52	4	5	A = 5 »Ton kurz«
53	5	3	D = A »Ton variabel«
54	1	9	A4 = 1
55	1	1	A4 = 0
56	2	1	2 ms
57	1	9	A4 = 1
58	1	1	A4 = 0
59	2	1	2 ms
5A	1	9	A4 = 1
5B	1	1	A4 = 0
5C	2	0	1 ms
5D	B	4	Dmal 04
5E	1	0	Dout 0
5F	E	0	Return

4F 93 45 53 19 11 21 19 11 21 19 11 20 B4 10 E0

Seite 5: Unterprogramm Tonausgabe

Adresse	Befehl	Daten	Kommentar
60	2	3	Warte 10 ms »Warte S1«
61	C	E	S1 = 1?
62	3	2	Springe -2
63	2	3	Warte 10 ms
64	C	C	S1 = 0?
65	3	1	Springe -1
66	E	0	Return
67	F	F	-
68	2	3	Warte 10 ms »Warte S2«
69	C	F	S2 = 1?
6A	3	2	Springe -2
6B	2	3	Warte 10 ms
6C	C	D	S2 = 0?
6D	3	1	Springe -1
6E	E	0	Return
6F	F	F	-

23 CE 32 23 CC 31 E0 FF 23 CF 32 23 CD 31 E0 FF

Seite 6: Unterprogramme Warte S1 und Warte S2

Adresse	Befehl	Daten	Kommentar
70	C	C	S1 = 0? »Tasteneingabe“
71	3	1	Springe -1
72	4	0	A = 0
73	5	4	Port = A
74	2	3	Warte 10 ms
75	C	E	S1 = 1?
76	3	2	Springe -2
77	C	F	S2 = 1?
78	E	0	Return
79	C	C	S1 = 0?
7A	3	3	Springe -3
7B	7	1	A = A + 1

Adresse	Befehl	Daten	Kommentar
7C	2	3	Warte 10 ms
7D	C	C	S1 = 1?
7E	3	1	Springe - 1
7F	3	C	Springe -12

CC 31 40 54 23 CE 32 CF E0 CC 33 71 23 CC 31 3C

Seite 7: Unterprogramm Tasteneingabe

Befehlstabelle

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

Impressum

www.elo-web.de
Autor: Burkhard Kainka

ISBN 978-3-645-10104-2

Produziert im Auftrag der Firma Conrad Electronic SE, Klaus-Conrad-Str. 1, 92240 Hirschau

Elektrische und elektronische Geräte dürfen nicht über den Hausmüll entsorgt werden! Entsorgen Sie das Produkt am Ende seiner Lebensdauer gemäß den geltenden gesetzlichen Vorschriften. Zur Rückgabe sind Sammelstellen eingerichtet worden, an denen Sie Elektrogeräte kostenlos abgeben konnen. Ihre Kommune informiert Sie, wo sich solche Sammelstellen befinden.

CE

Dieses Produkt ist konform zu den einschlägigen CE-Richtlinien, soweit Sie es gemäß der beiliegenden Anleitung verwenden. Die Beschreibung gehört zum Produkt und muss mitgegeben werden, wenn Sie es weitergeben.