Smartcard-Kode-Schloß

Zweites Leben für Chipkarten

Chipkarten lesen und beschreiben ist eine interessante Sache, nutzbringender ist sicherlich eine echte, aus dem Leben gegriffene Applikation. Unter den zahllosen Varianten zum Thema “Elektronisches Kodeschoß” hier eine außergewöhnliche, die eine neuprogrammierte Chipkarte als Schlüssel verwendet.

Entwurf von Patrick Gueulle

Die Applikation bietet einen hervorragenden Einstieg in die Chipkarten-Technik. Grundlage des Systems ist ein Mikrocontroller mit einem elektrisch löschbaren Programmspeicher (EEPROM) wie dem PIC16C84, der mit einem kinderleicht zu modifizierenden BASIC-Programm gefüttert wird.

LEARNING BY PROGRAMMING

Das Smartcard-Kodeschloß nutzt die schier unendliche Anzahl von möglichen Kodes, die in einer Smartcard verborgen sind. Eine einfache (entwertete) Telefon- oder Waschstraßenkarte identifiziert sich durch eine einzigartige Referenznummer, die zwar einfach aus dem Speicher der Karte gelesen, nicht aber geändert werden kann. Die 96 bit

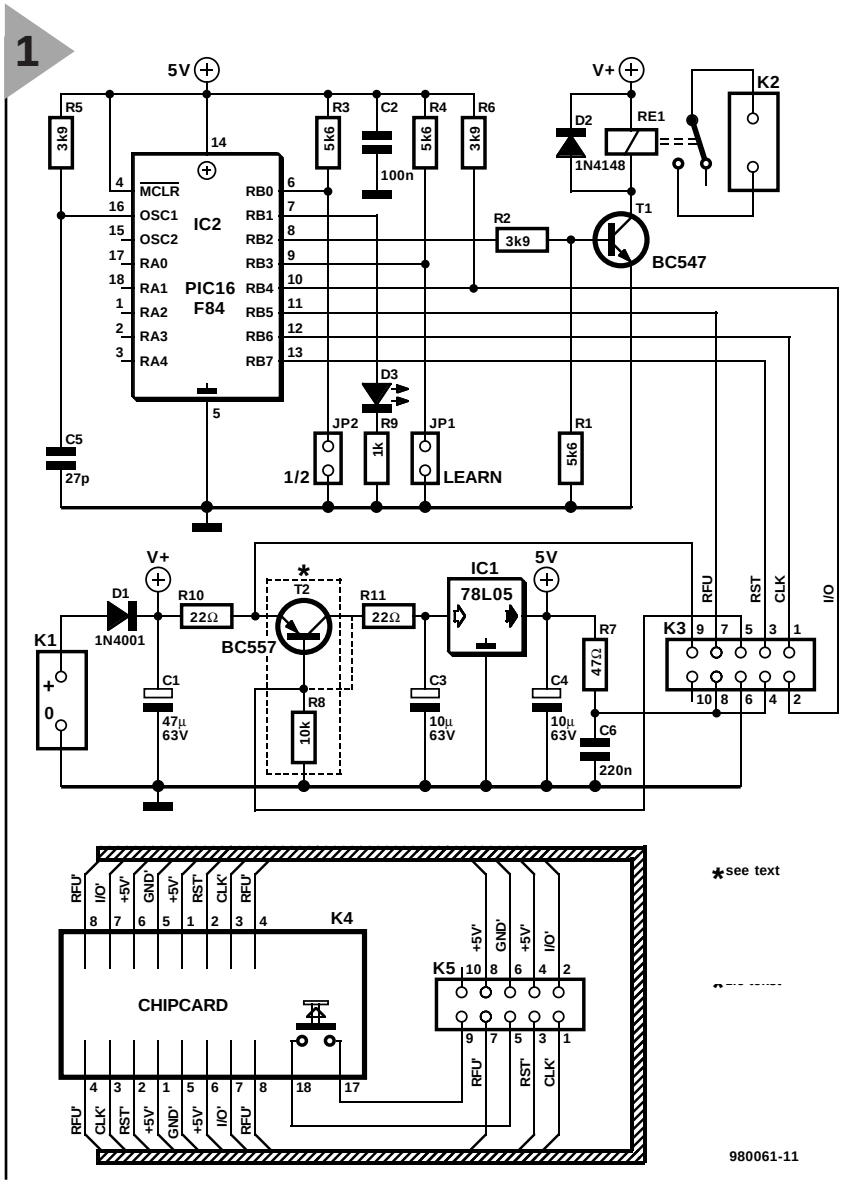


Bild 1. Die Schaltung des Chipkarten-Kodeschlosses besteht im wesentlichen aus einem PIC-Mikrocontroller und dem Kartenleser.

breite Referenznummer erlaubt die Vergabe von nahezu 80 Trillionen ($80 \cdot 10^{18}$!) verschiedenen Karten. Um diesen fast unaussprechlich großen Vorrat von verfügbaren Smartcards für ein Kodeschloß zu nutzen, benötigt man lediglich ein Lesegerät, das in der Lage ist, den Kode einer oder besser mehrerer leerer Karten zu lesen und in einem Speicher unterzubringen.

Ist diese Lernphase erst einmal absolviert, kann der Kartenleser zum Beispiel ein Relais ansteuern und damit eine Aktion auslösen, wenn eine der "gelernten" Chipkarten eingesteckt wird.

Alternativ lassen sich bestimmte Karten mit dem gleichen ID-Kode zu "personalisieren". Dies ist auch mit Karten möglich, die nicht für Telefonanwendungen vorgesehen waren, beispielsweise Karten für eine Autowaschstraße, Parkkarten, Pay-TV-Karten und so weiter. Eine leere Karte einer Autowaschstraße für beispielsweise 24 Waschungen enthält 136 bits mit einer Null. Mit einem Chipkarten-Leser/Schreiber wie dem in Elektor 9/97 vorgestellten lassen sich die Bits einfach und individuell auf Eins setzen.

Obwohl diese beiden Varianten im Vordergrund stehen dürften, gibt es noch die Möglichkeit, neue, unbeschriebene Karten zu kaufen. Soweit dem Autor bekannt ist, sind alle diese Karten nicht völlig leer, sondern enthalten eine Fabrikationsnummer in den ersten 96 dafür reservierten Speicherplätzen. Bei 256-bit-Karten weisen die nächsten 160 bits Nullen auf und können frei geändert werden. Das

```
'DUAL smart card lock with PIC16F84
'copyright (c) 1998 Patrick GUEULLE
symbol f = B0
symbol g = B1
symbol e = B2
symbol r = B3
symbol BLINK = B4
symbol h = B5
 DIRS = %11100110
 PINS = %00000010
 high 7
 high 6
 low 6
 low 7
 high 6
 low 6
 high 5
 if PIN3 = 0 then learn
 if PIN3 = 1 then check
learn:
 for f = 0 to 31
 e = 0
 for g = 0 to 7
 e = e * 2
 e = e + PIN4
 high 6
 low 6
 next g
 h = 32 * PIN0
 h = h + f
 write h,e
 next f
 PINS = 0
 BLINK = 50
 goto led
check:
 for f = 0 to 31
 e = 0
 for g = 0 to 7
 e = e * 2
 e = e + PIN4
 high 6
 low 6
 next g
 read f,r
 if e <> r then other
 goto cont
other:
 h = f + 32
 read h,r
 if e <> r then error
cont:
 next f
 PINS = %00000110
 goto forever
error:
 PINS = 0
 BLINK = 255
 goto led
led:
 for f = 0 to 10
 high 1
 pause BLINK
 low 1
 pause BLINK
 next f
forever:
 goto forever
```

Bild 2. PBASIC-Listing des PIC-Programms vor der Kompilierung.

Smartcard-Kodeschloß kann, abhängig von der konkreten Anwendung, ein oder zwei solcher "fälschungssicherer" Karten oder eine beliebige Anzahl identischer, unter bestimmten Voraussetzungen kopierbarer Karten erkennen.

GANZ IN BASIC: DER PIC16C84

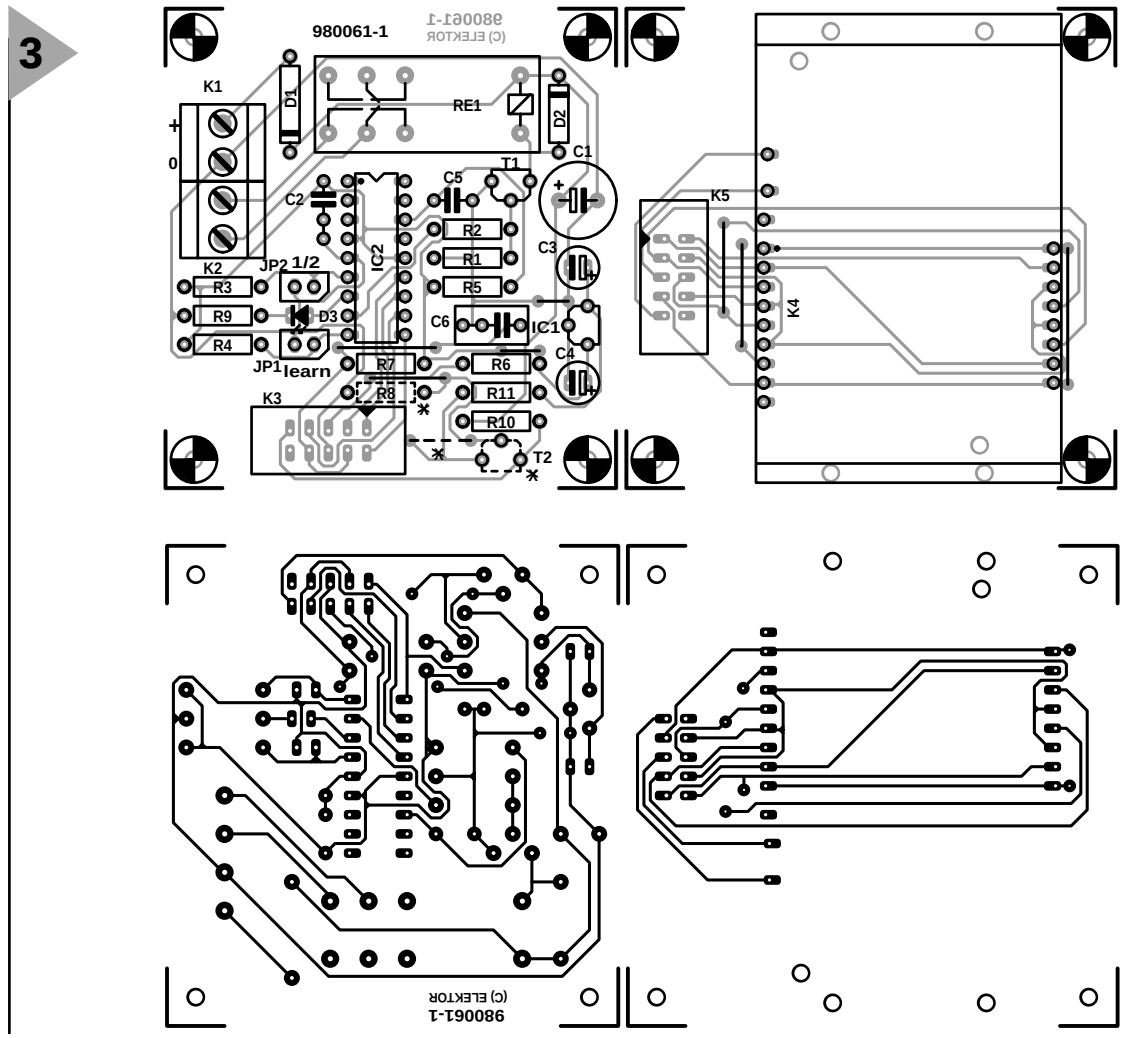
Das Überraschende an der Schaltung in **Bild 1** ist der angesichts der komplizierten Funktionen einfache Aufbau. Kein Wunder, denn der eingesetzte Mikrocontroller, ein PIC16C84, enthält alle notwendigen Ingedrenzien eines vollständigen Mikrocontrollersystems. So kann zum Beispiel das 64 byte große Daten-EEPROM, ein nichtflüchtiger Speicher, zwei 256 bit breite *card images* festhalten, und zwar über zehn Jahre und beliebig oft änderbar.

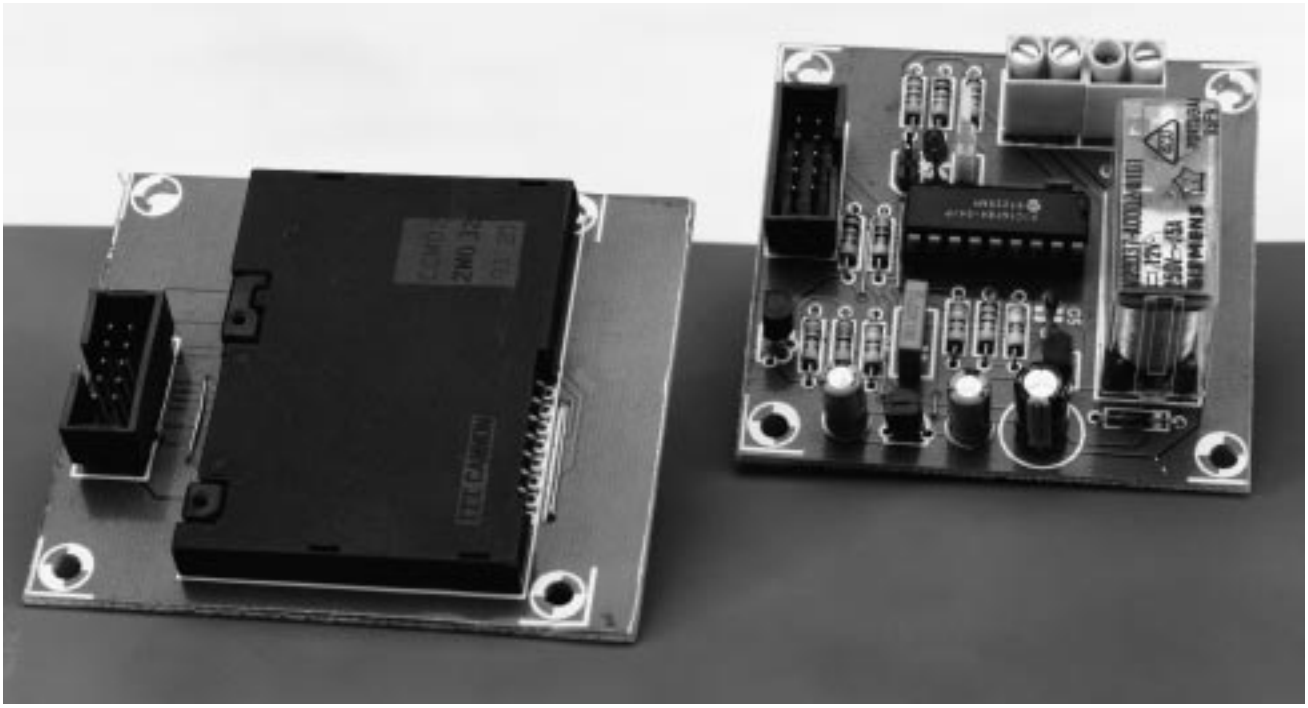
Da die Schaltung keinen nennenswerten Strom benötigt, wenn keine Karte im Leseadapter steckt, läßt sie sich mit einer normalen 9-V-Blockbatterie betreiben. Es spricht aber auch nichts dagegen, ein Steckernetzteil oder eine aufladbare Batterie beispielsweise einer Alarmanlage zur Stromversorgung heranzuziehen. Welche Stromquelle man auch wählt, ein 78L05-Spannungsregler stabilisiert die Betriebsspannung für den PIC auf 5 V, während die Relaisspule die ausreichend hohe Eingangsspannung nutzt. Klar, das die Einfachheit der Schaltung erst durch den PIC-Mikrocontroller und sein Programm ermöglicht wird. Das Steuerprogramm ist in einem BASIC-Dialekt namens PBASIC verfaßt. PBASIC wurde ursprünglich von Parallax für den Mikrocomputer BASIC-Briefmarke entwickelt. Obwohl es leicht möglich gewesen wäre, das in **Bild 2** dargestellte Programm einfach zu einer mit einem Leseadapter verbundenen BASIC-Briefmarke 1 herunterzuladen, hat der Autor dieser (teuren) Versuchung widerstanden und wählte statt dessen den Weg über die Komplierung des Programms. Dazu stehen diverse PBASIC-Compiler verschiedener renommierter Firmen wie *Forest Electronic Developments* oder *Micro Engineering Labs* zur Verfügung, aber auch brauchbare Public-domain-Software. Die leistungsfähigsten dieser Programme produzieren einen Objektkode, der sich durchaus mit den Ergebnissen eines Assemblers messen kann, die einfacheren Ausführungen sind im Grunde nicht mehr als Interpreter.

Dies soll aber nicht sagen, daß der Begriff Interpreter hier unangebracht wäre, da das BASIC-Programm in Form von Tokens (ein paar Bytes pro Befehl) im Speicher untergebracht wird. Anschließend interpretiert ein residentes "Softwärchen" den Befehl Bit für Bit. Dieser Vorgang wird als – nein, nicht interpretieren, sondern – *tokenising* bezeichnet.

Der für dieses Projekte verwendete Compiler lädt die Firmware und das tokenisierte BASIC in einen gewöhnlichen PIC16F84. Dazu werden Dreiviertel des zur Verfügung stehenden Speicherplatzes gebraucht. Im Gegensatz dazu greift die BASIC-Briefmarke 1 auf einen maskenprogrammierten Interpreter im lesegeschützten EPROM eines OTP-PICs 16C56 (one time programmable) zurück und bringt das Anwenderprogramm in

Bild 3. Layout und Bestückung der zweiteiligen Platine für das Chipkarten-Kodeschloß.





einem separaten EEPROM vom Typ 93LC56 unter.
Wie auch immer, es bleiben immer 256 Byte Raum für das Applikationsprogramm. Das scheint sehr wenig, um vernünftig damit zu arbeiten, bietet allerdings dank des kompakt strukturisierten tokenisierten BASICs ausreichend Platz. Sicher ist in der Entwicklungsphase eine BASIC-Briefmarke praktischer, die Kosten sprechen aber deutlich für einen "leeren" PIC, wenn Hard- und Software erst einmal "stehen".
Es war bei der Programmentwicklung eine große Hilfe, einen Compiler zu verwenden, der sowohl mit der BASIC-Briefmarke und dem PIC16F84 kompatibel ist. Insbesondere bei den bei der Assemblerprogrammierung umständlichen Zugriffen auf das EEPROM ergaben sich deutliche Vorteile dieser Lösung.
Es geht das Gerücht, daß interpretiertes BASIC nicht gerade Geschwindigkeitsrekorde bricht. Dies scheint sich aber hier als Pluspunkt erweisen, denn die Software benötigt keinerlei Verzögerungsschleifen, um ein bestimmtes Timing einzuhalten.
Der PIC wird durch ein RC-Netzwerk mit 2 MHz getaktet und benötigt so weniger als eine Sekunde, um die 256 Byte einer synchronen Chipkarte nach dem deutschem oder französischem Protokoll für Daten/Befehlsaustausch zu lesen. Eine zusätzliche Reset-Möglichkeit erlaubt den Einsatz von Telefon-Chipkarten fast jeden Standards.

REALISATION

Beim Aufbau eines elektronischen Kodeschlosses gleich welchen Prinzips sind immer eine Reihe von elementaren Sicherheitsmaßnahmen zu beachten. Es ist sicherlich elegant, aber auch (zu) leichtsinnig, alle Bauteile als SMDs auszuführen, um sie zusammen mit dem Leseadapter auf einer kompakten Platine unterzubringen. Selbst einige sich als "professionell" bezeichnenden Leseterminals bekannter Hersteller sind nach dieser Machart konstruiert. Durch einfaches Abschrauben kann man das Terminal entfernen und die offenliegenden Kabel zum Beispiel zu einem Türöffner verbinden, um das Schloß zu überlisten. Aus diesem Grund haben wir eine zweiteilige Platine (**Bild 3**) entworfen, die es erlaubt, die Elektronik an einem sicheren Platz getrennt vom Terminal zu montieren.
Beim offen zugänglichen Terminal kann man die ISO- und AFNOR-Kontakte verbinden, um den Gebrauch beider Kartentypen zu erlauben. Die AFNOR-Pinbelegung ist übrigens mittlerweile auch außerhalb Frankreichs gebräuchlich. Die gestrichelt eingezeichneten Bauteile R8 und T2 erlauben den Einsatz von Kartenadaptern mit Arbeitskontakten (normally open, NO) oder mit Ruhekontakten (normally closed, NC). Im ersten Fall werden R8 und T2 weggelassen und die Bohrungen von Basis und Kollektor mit einer Drahtbrücke versehen. Diese Verbindung ist gestrichelt gezeichnet. Beim NC-Adapter montiert man die beiden Bauteile (und läßt selbstredend die Drahtbrücke weg).

Stückliste

Widerstände:
R1,R3,R4 = 5k6
R2,R5,R8 = 3k9
R7 = 47 Ω
R8 = 10 k
R9 = 1 k
R10,R11 = 22 Ω

Kondensatoren:
C1 = 47 µ/63 V stehend
C2 = 100 n
C3,C4 = 10 µ/63 V stehend
C5 = 27 p
C6 = 220 n

Halbleiter:
D1 = 1N4001
D2 = 1N4148
D3 = LED, rot, 3 mm, low-current
T1 = BC547
T2 = BC557
IC1 = 78L05
IC2 = PIC16F84 (EPS 986511-1)

Außerdem:
K1,K2 = 2polige Platinenanschlußklemme, RM5
Re1 = 12-V-Relais 1um (Siemens V23037-A2-A01)
JP1,JP2 = 2poliger Pfostenverbinder mt Jumper
K3,K5 = 2·5poliger Pfostenfeldverbinder mit Schutzkragen und Kabelverbinder
K4 = Chipkarten-Leser (ITT Canon CCM02 2NO 93 20)
10poliges Flachbandkabel
Platine EPS 980061-1
Software (prog. PIC) EPS 986511-1
(siehe Serviceseiten in der Heftmitte)

Die Verbindung zwischen den beiden Platinen übernimmt ein 2·5poliges Flachbandkabel, ausgestattet mit entsprechenden Pfostenfeld- und Kabelverbindern. Dank der nicht überragenden Signalfrequenz von etwa 500 Hz dürften Kabellängen von 50 cm kein Problem sein.
Das 9...15-V-Steckernetzteil wird zur Spannungsversorgung ebenso an eine massive Platinenanschlußklemme (K1) angeschlossen wie die zu schaltende Last (an K2). Das Relais verbindet diese beiden Kontakte nur, wenn die richtige Karte einsteckt wird. Die LED informiert über die korrekte Abwicklung der einzelnen Prozeßschritte und ist im "normalen" Betrieb überflüssig. Zwei Jumper entscheiden über zwei Betriebszustände: Lern- (JP1) und normaler Betrieb (JPs). Die Jumper sind ebenfalls ausschließlich in der Anlaufphase nötig, wenn neue Karten gelernt werden sollen.
Der programmierte PIC-Controller ist unter der Nummer EPS 986511-1 im Service erhältlich. Er sollte erst nach gründlicher Kontrolle (Sicht-Check, Prüfen der Betriebsspannungen) der ansonsten fertig aufgebauten Hardware in seine Fassung gesteckt werden.

In Gebrauch

Die Vorbereitung des Kodeschlosses auf die Chipkarten ist fast noch einfacher als der Aufbau der Schaltung. Zunächst muß dem PIC beigebracht werden, auf eine bestimmte Karte zu reagieren. Dazu setzt man den Jumper JP1 in die LEARN-Stellung, so daß Portleitung RB3 des PICs auf Masse liegt, und schiebt die Karte in den Leser. Die LED muß aufleuchten, um zu zeigen, daß Betriebsspannung vorhanden ist. Nach ungefähr einer Sekunde beginnt die LED zu blinken und zeigt, daß die Karten-ID im EEPROM gespeichert ist. Die ID verharrt dort monatelang, selbst wenn die Versorgungsspannung wegfällt, bis eine anderer Karte eingeschoben wird. Die LED verlischt nun, so daß man gefahrlos die nun als Schlüssel registrierte Karte herausziehen kann. Den LEARN-Jumper entfernt und die Karte wieder eingeschoben, muß die LED aufleuchten und das Relais innerhalb einer Sekunde anziehen. Dieser Zustand ist stabil, solange die Karte im Adapterschlitz steckt.
Probieren Sie es mit einer anderen Karte aus. Selbst wenn diese vom Schlüssel kopiert ist und den gleichen Inhalt aufweist, blinkt die LED langsam und verlischt nach kurzer Zeit völlig. Das Relais bleibt natürlich in seiner Aus-Position.
Den ganzen Vorgang kann man mit gestecktem Jumper 1/2 wiederholen, um einen zweiten Schlüssel zu definieren. Soll das System nur einen Schlüssel erkennen, muß er in beiden Jumperpositionen mit der gleichen Karte programmiert werden, um zu vermeiden, daß der undefinierte Zustand einer EEPROM-Hälfte die Systemsicherheit reduziert.

Karten kodieren

Obwohl es in den meisten Anwendungen ausreichend sein dürfte, daß nur zwei verschiedene Chipkarten das Kodeschloß auslösen, sind doch Fälle denkbar, in denen eine große Zahl von Autorisierten ein und dieselbe Tür öffnen darf, etwa in einem Büro/Labor mit Zugangskontrolle, einem Clubhaus oder einem Appartementblock. Dann kommt es darauf an, eine Reihe von Karten zu organisieren, die ursprünglich den gleichen Verwendungszweck hatten, zum Beispiel Karten, die von manchen Autowaschstraßen verkauft werden. Viele der entwerteten Karten werden von ihren Besitzern einfach weggeworfen.
Eine Smartcard mit beispielsweise 24 Einheiten hat folgende 256 bits zum Inhalt:

```
1000 1000 1000 0000 0010 0000 0000 0010
0011 1100 0111 0101 1000 0010 0010 0100
1010 0001 0000 0000 0000 0000 0000 0001
0000 0000 0011 1111 1111 1111 1111 1111
1100 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

In den ersten drei Zeilen, die nicht modifiziert werden können, ist der Name des Kartenherstellers kodiert. Der Namenskode ist in auf allen 24-Einheiten-Karten identisch, stimmt aber nicht mit 12-Einheiten-Karten überein (die übrigens genauso geeignet sind). Nun folgen in der vierten Zeile zehn Bits mit dem Inhalt Null, anschließend 24 Einsen, die die entwerteten 24 Einheiten repräsentieren. Die abschließenden 126 Nullen spielen in dieser Applikation keine Rolle.
Es liegt auf der Hand, daß man das Kodeschloß nicht auf eine derartige Karte einstellt, denn dann könnte jeder mit einer gewöhnlichen entwerteten 24-Einheiten-Waschstraßenkarte das Schloß öffnen.
Statt dessen versieht man die Karte(n) mit einem persönlichen Kode, um die Anzahl der autorisierten Personen zu begrenzen. Ein solcher Kode kann leicht mit der Software ELEKT1G und dem dazugehörenden Chipkarten-Leser/Schreiber aus Elektor 9/97 erstellt werden.
Das folgende Beispiel zeigt, wie nach dem Lesen der Karte jede der 126 Nullen in eine Eins verwandelt werden kann, indem man einfach die "+"-Taste des Keyboard drückt, wenn der Cursor auf dem entsprechenden Bit steht.

Leertaste:	*Folgendes Bit lesen (auto-repeat).*
+-Taste:	*Eine Eins an die gegenwärtige Cursorposition schreiben*
ESCape-Taste:	*Ende*

```
1000 1000 1000 0000 0010 0000 0000 0010
0011 1100 0111 0101 1000 0010 0010 0100
1010 0001 0000 0000 0000 0000 0000 0001
0011 0000 0011 1111 1111 1111 1111 1111
1100 0000 0000 0000 0000 0000 0000 0000
0000 0000 1010 0000 0000 0000 0000 0000
0000 0000 0101 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

(Warnung: Vpp ausschalten!)

Für alle Karten, die gleichfalls das Schloß öffnen sollen, wiederholt man die Prozedur, und mit einer dieser Karten programmiert man das Kodeschloß.
Das Ganze funktioniert auch mit leeren Karten des Typs GPM256 oder ähnlichen mit einem EEPROM von 256 bit. Ob die Karte neu oder gebraucht (abgelaufen, leer) ist, nur die ersten 96 bit sind durch den Hersteller festgelegt, die verbleibenden 160 bit sind frei verfügbar. Hier ein Beispiel einer beliebigen Karte aus einer laufenden Produktionsserie (batch):

```
0000 0000 1000 0000 0000 0000 0000 0000
0100 1011 0000 0000 0000 0000 0000 0000
1111 1111 1111 0000 0000 1111 1111 1111
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

Das Programm ELEKT1G kann ohne Probleme für solche Karten gebraucht werden, um eine beliebige Zahl von Bits in den letzten fünf Zeilen auf Eins zu setzen.

```
0000 0000 1000 0000 0000 0000 0000 0000
0100 1011 0000 0000 0000 0000 0000 0000
1111 1111 1111 0000 0000 1111 1111 1111
1111 1111 0000 0000 0000 0000 0000 0001
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 1100 0011 0000 0000 0000
1000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 1110 0010
```

(Warnung: Vpp ausschalten!)

Bedenken Sie, daß eine solche Karte kopiert werden kann, aber nur dann, wenn jemand eine Karte mit dem gleichen 96 bit breiten Identifikationskode des gleichen Herstellers, also aus derselben Produktionsserie verwendet. Und dies dürfte sehr unwahrscheinlich sein.

(980061)rg