Kartenspiele

Gunther Zielosko

1. Grundlagen

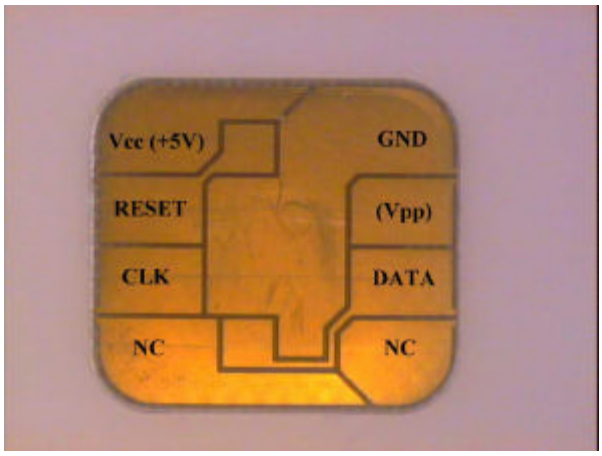
1.1. Allgemeines zu Chipkarten

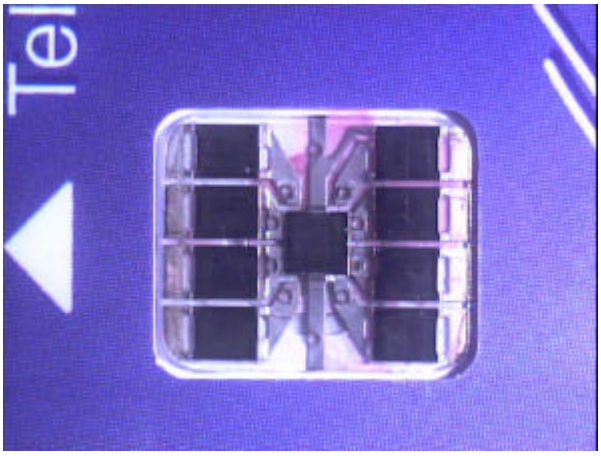
Jeder hat sie, die kleinen Plastikkarten mit den geheimnisvollen Goldkontakten, sei es als Geldkarte für das Girokonto, als Versichertenkarte für den Arztbesuch oder als Telefonkarte. Natürlich macht das den Elektronik-Anwender neugierig, BASIC-Tiger®-Besitzer sind da keine Ausnahme, sie haben sogar bessere "Karten", wenn es um Experimente mit diesen Datenträgern geht. Sollten Sie allerdings den Wunsch haben, Ihre Geldkarte unter Umgehung der Bank wieder aufzuladen, wird Ihnen dieser Artikel auch nicht weiterhelfen, dennoch werden wir Interessantes und Nützliches mit Chipkarten tun. Doch zunächst ein paar Informationen zum Aufbau dieser Informationsträger.

In den so unscheinbaren flachen Chipkarten befindet sich echte High-Tech-Elektronik. Äußerlich sichtbar sind eigentlich nur die manchmal unterschiedlich aussehenden Kontaktflächen (Bild 1).

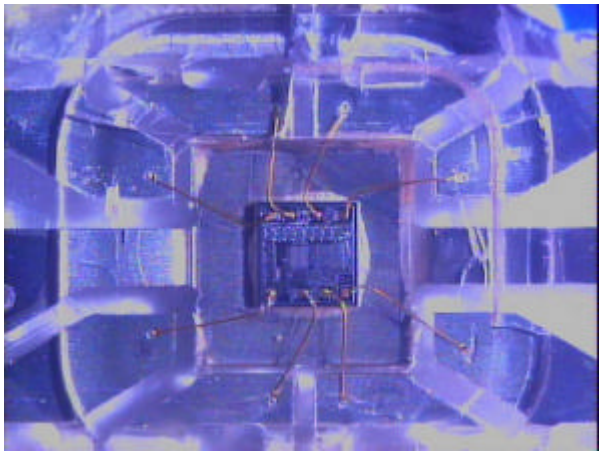
(Bild 1: eine typische Telefonkarte)

(Bild 2: standardisierte Kontaktbelegung)





(Bild 3: der eingebaute Chip)



(Bild 4: Chip mit seinen Bonddrähten)

Trotz unterschiedlicher Geometrie haben aber alle immer an der gleichen Stelle ihre Kontaktpunkte, zur Zeit acht, von denen 2 meist nicht angeschlossen und manchmal gar nicht vorhanden sind. Die Kontaktbelegung zeigt Bild 2.

Direkt unter der Mitte des Kontaktbereiches ist "rückwärts" ein Chip untergebracht, der über Bonddrähte mit den einzelnen Kontakten verbunden ist. All dies ist in "Flachbauweise" in die knapp einen Millimeter dicke Karte eingebettet (Bilder 3 und 4). Der eingebaute Chip hat mehrere Funktionsgruppen, so z.B. mindestens ein serielles Interface für die Kommunikation mit dem Kartenleser, ein EEPROM zum nichtflüchtigen Speichern der Daten, Module zum Aufbereiten der Versorgungsspannungen, Schutzeinrichtungen gegen elektrostatische Elektrizität. Aber es gibt auch Karten mit kompletten Mikrorechnersystemen, also Prozessor, RAM, ROM, EEPROM usw.

Je nach Aufgabe existieren viele Varianten von Chipkarten, einfachere Versionen benutzen etwas Logik und ein EEPROM zur Speicherung von Daten, kompliziertere haben zusätzlich einen kompletten Mikrocontroller. Zu den sogenannten synchronen Karten gehört die Telefonkarte, ein Vertreter der asynchronen Chipkarten ist die Geldkarte. Trotz der Unterschiede in der Kommunikation ist das Prinzip immer das gleiche, nach dem Einschieben der Karte in einen Kartenleser wird ein Kontakt betätigt, der die Betriebsspannung von +5 V an den Anschluß Vcc der Karte bringt und die Auswerteelektronik startet. Die externe Elektronik und ggf. Software sorgt danach für einen Reset-Impuls am Anschluß RST (hier beginnen schon die Unterschiede der Kommunikation). Der Clockanschluß CLK der Chipkarte wird nun mit einer Taktfrequenz belegt, das Kartenchip liefert bei jedem Taktimpuls am Datenpin I/O ein Bit der gespeicherten Information ab.

1.2. Chipkarten für den „Eigenbedarf“

Alle bisher erwähnten Chipkarten sind mehr oder weniger „fremdbestimmt“, die meisten Daten sind geschützt, was bei Geld- oder Versicherungskarten einleuchtet. Aber es gibt mittlerweile auch Chipkarten, die völlig frei gelesen und sogar beschrieben werden können. Auch solche Chipkarten haben EEPROM’s verschiedener Größen und kommunizieren über eine normale I^2C-Schnittstelle mit ihrer Umwelt.

Diese Chipkarten sind genau das Richtige für uns, bis auf eine Kontaktiereinrichtung und ein paar simple diskrete Bauteile braucht es kaum Hardwareaufwand, um mit der Karte zu arbeiten. Wie das funktioniert, zeigt das nächste Kapitel. Vorher wollen wir aber ein paar Überlegungen für die Nutzung von I^2C-Chipkarten mit dem BASIC-Tiger® anstellen. Was könnte man mit solchen Speicherkarten am BASIC-Tiger® anfangen? Ist es sinnvoll, den eigentlich reichlichen Speicher (sowohl RAM als auch EEPROM!) des BASIC-Tigers® extern aufzurüsten? Immerhin hat bereits die kleinste Version des Tigers im Vergleich zu den verfügbaren Chipkarten einen riesigen EEPROM-Speicher, der ja auch zur nichtflüchtigen Speicherung von Daten genutzt werden kann. Hier ein paar Vorschläge, wie man externen Speicher in Form von Chipkarten sinnvoll nutzen kann:

Parametereingabe für ein existierendes Programm

BASIC-Tiger®-Anwender sind bei Parameteränderungen am Programm immer wieder mit der Situation konfrontiert, die BASIC-Tiger®-Software aufrufen zu müssen, ggf. den Tiger aus seiner Schaltung zu entfernen und das neue Programm einladen zu müssen. Wenn Sie beispielsweise einen Funktionsgenerator mit selbsterstellten Funktionen oder einen Patterngenerator mit ständig zu variierenden Bitmustern aufgebaut haben, könnte die Funktion oder das Bitmuster von der Chipkarte kommen, die Sie bei Bedarf einfach wechseln. Das ist vermutlich für viele Anwender interessant.

Zugangskontrolle

In der heutigen elektronisch gesicherten Welt kennen Sie Chipkarten als Zugangskontrolle. Wie wäre es, selbst ein solches System mit dem BASIC-Tiger® aufzubauen? Beschreiben Sie Ihre Chipkarte mit Daten Ihrer Wahl, unter anderem z.B. mit einem Codewort. Das BASIC-Tiger®-Programm liest die eingesteckte Chipkarte und kontrolliert speziell dieses Codewort. Das öffnet eine Tür, startet Ihren PC oder ein beliebiges anderes Gerät. Wenn Sie mehrere Karten benutzen, können die unterschiedlichen Daten der Karteneigentümer z.B. zur Zeiterfassung dienen (wer hat wann telefoniert oder war am Computer?) Genauso ist ein professionelles Zeit- und Anwesenheits-Erfassungssystem für kleine Firmen möglich.

Daten eines Meßsystems erfassen

Sie haben ein Stand-alone-Meßsystem aufgebaut, das irgendwo fleißig jede Stunde die Temperatur oder etwas anderes mißt. Da ist es praktisch, wenn die Daten auf Knopfdruck auf eine Chipkarte gespeichert werden können, die Sie dann mitnehmen und an einem anderen Gerät auslesen können.

2. Hardware

2.1. Bauteile

Als erstes brauchen Sie die Chipkarten selbst. Man bekommt sie z.B. bei Conrad Electronic (www.conrad.de) in verschiedenen Speichergrößen:

2Kbit	für kleinere Anwendungen ausreichend (Bestell-Nr. 96 78 15-11)
16Kbit	mit unserem EEPROM 24LC16B (Bestell-Nr. 97 29 24-11)
64Kbit	für größere Projekte (Bestell-Nr. 97 29 67-11)

Wer möchte, kann auch mit dem diskreten I^2C-EEPROM 24LC16B experimentieren:

24C16 (Bestell-Nr. 16 09 97-11)

Wer sich intensiver mit dem EEPROM 24LC16B in Chipkarten beschäftigen will, kann bei Conrad auch ein Datenblatt bekommen:
http://www.produktinfo.conrad.de/datenblaetter/900000-999999/972924-da-01-en-I2C-Chipcard_16KBIT.pdf

Nun ist eine Kontaktiereinrichtung für die Chipkarten erforderlich. Am einfachsten ist der Kauf eines handelsüblichen Bauteiles, wie es z.B. von der Firma ELV Elektronik angeboten wird (Bilder 5 und 6, hier mit einer eingesteckten Telefonkarte):

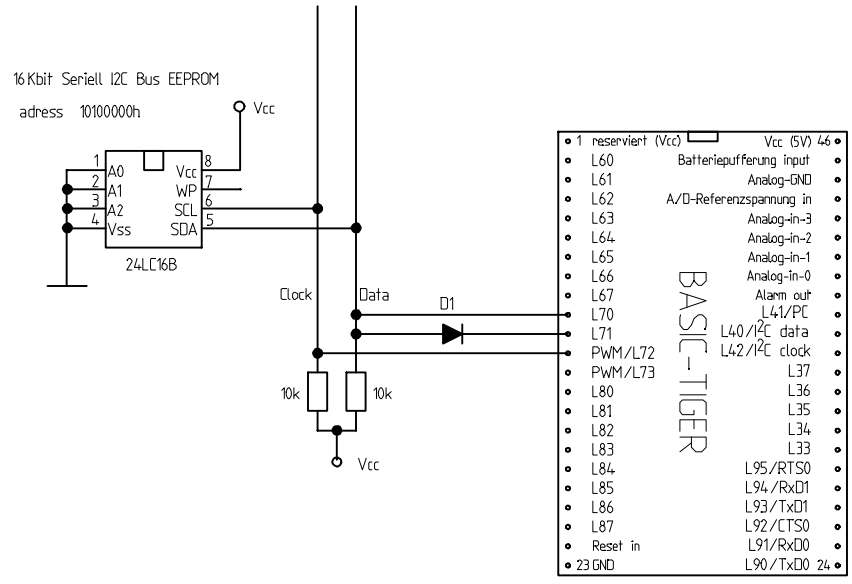
ELV Elektronik
Postfach
26787 Tel.: 0491 600888 Internet: http://www.elv.de Best.-Nr. 50-160-92

(Bild 5: Kartenadapter von Amphenol)

(Bild 6: Rückseite mit Lötkontakten)

2.2. Schaltung

Die Verdrahtung des Chipkartenadapters ist denkbar einfach. Prinzipiell werden außer Betriebsspannung und Masse nur zwei Leitungen benötigt, um ein EEPROM (und unsere Chipkarte ist nichts anderes!) über die I²C-Schnittstelle anzusteuern. Die I²C-Schnittstelle in der Version 4 des BASIC-Tigers® wird in unserem Fall nicht automatisch von den Pins L1-0/I2C data und L1-2/I²C clock abgeleitet, sondern aus den per BASIC-Funktion I²C-SETUP festgelegten 3 Pins des BASIC-Tigers® (siehe I²C-Beispielprogramme I²C_SETUP). In unserem Beispiel werden die Leitungen L70, L71 und L72 benötigt, die wie folgt mit einem EEPROM 24LC16B oder baugleichen Typen verschaltet werden. Die beiden senkrechten Leitungen Clock und Data sind die beiden Standardleitungen des I²C-Bussystems, hier können bei Bedarf weitere I²C Bauelemente angeschlossen werden. Diese haben dann je nach Typ unterschiedliche Adressen, unser EEPROM hat die Typ-Adresse1010XXXXb.



(Bild 7 Anschluß eines 16Kbit-EEPROMs mit I²C-Schnittstelle an den BASIC-Tiger®)

Eigentlich wollten wir aber eine Chipkarte ansteuern. Ganz einfach, nur Vcc, Vss, DATA und CLK (in der Schaltung Bild 7 Data und Clock genannt) an die jeweiligen Kontakte des Chipkartenadapters legen (Bild 2 beachten!). Die nicht erwähnten Anschlüsse Vpp und Reset können bei dieser Applikation freibleiben, ebenso die zum Schalten der Betriebsspannung beim Einschieben einer Karte in den Adapter gedachten Anschlüsse.

3. Benutzung der Chipkarten

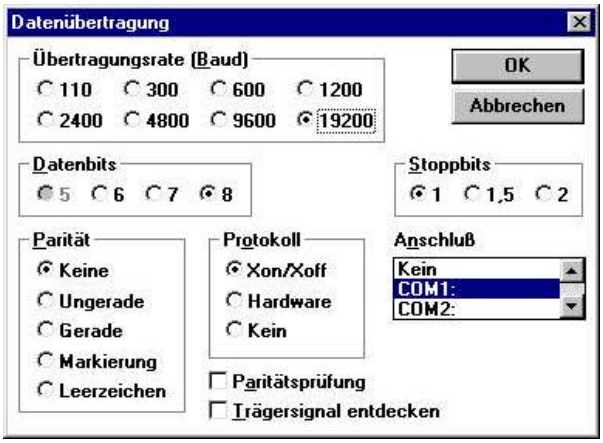
3.1. Zusammenspiel mit dem PC

In diesem Applikationsbericht können natürlich nicht alle Aspekte des Themas Chipkarten und BASIC-Tiger® erschöpfend behandelt werden. Wir beschränken uns auf die einfache Variante, I^2C-Chipkarten zu beschreiben und zu lesen. Die im Kapitel 3.2. vorgestellte Software besteht aus zwei BASIC-Tiger®-Programmen, die genau dies tun. Grundlage dafür ist das Beispielprogramm von Wilke I2C2416.TIG. Will man die 2048 Bytes aus einem EEPROM 24LC16B oder der entsprechenden Chipkarte lesen, reicht das Standard-Display des Plug-and-Play-Labs nicht mehr aus. Deshalb wurde hier einmal ein anderer Weg zur Anzeige großer Datenmengen beschritten. Die Speicherinhalte werden seitenweise (jeweils 256 Bytes) aus der Karte in den BASIC-Tiger® gelesen und dann über die serielle Schnittstelle in den PC übernommen. Dazu braucht man ein Terminalprogramm. Sowohl Windows 3.1 als auch Windows 95/98 haben solche Dienstprogramme. Sie können mehr oder weniger komfortabel Daten aus der seriellen Schnittstelle einlesen und anzeigen. Der Autor benutzt für die vorliegende Anwendung das Terminalprogramm TERMINAL.EXE von Windows 3.1, das ohne Einschränkungen auch unter Windows 95 läuft und zwar unabhängig, d.h. das diesem Applikationsbericht beigefügte Programm TERMINAL.EXE wird einfach irgendwohin kopiert und aufgerufen. Bevor man mit der Datenübertragung beginnen kann, müssen die Parameter der entsprechenden Schnittstelle eingestellt werden. Unter Einstellungen -> Datenübertragung kommt man zu dem Menü in Bild 8, hier werden die für die BASIC-Tiger®-Software gültigen Parameter eingestellt. Wenn Sie noch unter Einstellungen -> Terminal-Einstellungen ganz rechts unten die Puffergröße auf 300 Zeilen einstellen (Bild 9), können Sie den gesamten Inhalt der Chipkarte betrachten.

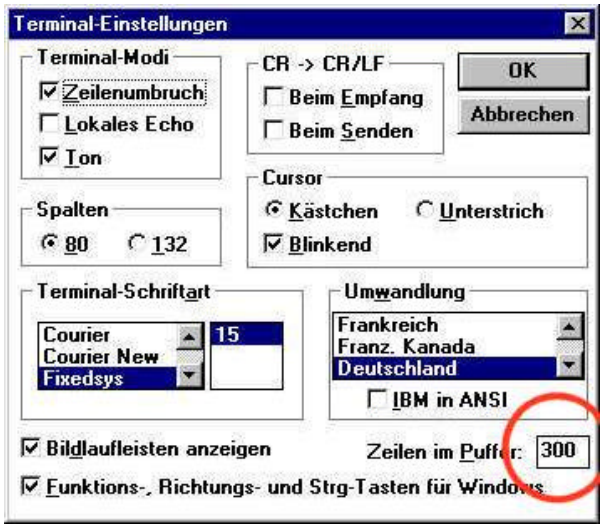
Im Programm EEPROM_L wurde die serielle Schnittstelle SER1 eingestellt, also dieselbe, die Sie auch zur Programmierung des BASIC-Tigers® nutzen. Damit entfällt das Umstecken des Schnittstellenkabels. Allerdings müssen Sie beim Programmieren des BASIC-Tigers® das Terminalprogramm schließen und umgekehrt, weil sie ja beide auf dieselbe Schnittstelle zugreifen.

Nach diesen Vorbereitungen ist das Terminalprogramm bereit, Daten vom BASIC-Tiger® aufzunehmen. Vor der Übertragung neuer Daten können Sie den Puffer des Terminalprogrammes (und damit die Datenanzeige auf dem Bildschirm) unter Bearbeiten -> Puffer löschen.

Im Ergebnis bekommen wir ein komplettes Hex-Dump des EEPROM-Inhalts. Die 8 Seiten (pages) des EEPROM's werden mit Zwischenraum sowohl als Hexadezimalwerte (links) als auch in Form der dazugehörigen ASCII-Zeichen ausgegeben (rechts).



(Bild 8 Parameter der seriellen Schnittstelle)



(Bild 9 Puffer auf 300 Zeilen einstellen)

3.2. BASIC-Tiger®-Software

Es werden zwei einfache Programme vorgestellt, die ganz auf das Lesen (EEPROM_L) und Schreiben (EEPROM_S) eines EEPROM's 24LC16B spezialisiert sind. Die Anzeige erfolgt dabei, wie schon erwähnt, mittels serieller Schnittstelle und Terminalprogramm TERMINAL.EXE auf einem PC.

```
'-------------------------------------------------------------------------------
' Name: EEPROM_L.TIG
' Der Inhalt einer Chipkarte mit EEPROM M24LC16 (2kByte) wird ueber den
' I2C-Bus in den BASIC-Tiger eingelesen. Danach werden die Daten als
' über die serielle Schnittstelle (hier SER1, derselbe Kanal wie beim
' Einlesen des Programms) an den PC uebertragen. Ein Terminal-Programm
' zeigt dann alle 2048 Bytes als Hex-Dump auf dem PC-Bildschirm an.
'
' Anbindung BASIC-Tiger -> Chipkarte (oder EEPROM)
' Vcc   -> Vcc
' Vss   -> Vss
' L70 BASIC-Tiger -> Data
' L72 BASIC-Tiger -> Clock
' L71 Kathode Diode -> Anode Diode -> Data
'
' Einstellungen des Terminalprogrammes
' 19200 Bd, keine Paritaet, Xon/Xoff, 1 Stopbit
'-------------------------------------------------------------------------------
user_var_strict                         ' strenge Variablen-Deklaration

#include DEFINE_A.INC                   ' Include-Dateien
#include UFUNC3.INC                     ' einbinden

#define M24LC16 0A0h                    ' Adresse des EEPROM Seite 0
#define N_PAGES 8                       ' Anzahl der Seiten = 8
ARRAY R$(8) OF STRINGS(256)             ' Array fuer die zu lesenden Strings
BYTE EE_ADR                             ' Variable fuer die Seiten-Adresse
```

```
'-------------------------------------------------------------------------------
TASK MAIN                               ' Beginn der Task Main

  LONG C, I, L, P, Q                    ' Variablen-Deklaration
  BYTE SP, A
  STRING C$(1), RA$(256), RL$(16)

  INSTALL_DEVICE #SER, "SER1B_K1.TDD",&       ' Serielle Schnittstellen
   BD_19_200, DP_8N, JA, BD_19_200, DP_8N, JA  ' einstellen
  USING "UH<2><2>   0.0.0.0.2"

  PRINT #SER, #1, "Inhalt der 8 Seiten der Chipkarte "

' I2C_SETUP  (pin, din, pout, dout, clout, spcnt)
  I2C_SETUP  (  7,   0,    7,    1,     2,     0)
  CALL EE_RD                            ' 8 Seiten lesen
    PRINT #SER,#1, "<13><10>";          ' leere Zeile

  FOR P = 0 TO N_PAGES-1                ' Anzeigen des Hex-Dumps fuer die Seiten
    RA$ = R$(P)                         ' eine Seite in RA$ schreiben
    FOR L = 0 TO 15                     ' 16 Zeilen
      RL$ = ""                          ' ASCII Zeichen in Hex-Dump umwandeln
      FOR C = 0 TO 15                   ' 16 Spalten
        PRINT_USING #SER, #1, ASC(MID$(RA$,L*16+C,1));" ";
        IF C = 7 THEN                   ' Zwischenraum nach 8 Bytes
          PRINT #SER, #1, "- ";
        ENDIF
        C$ = CHR$(ASC(MID$(RA$,L*16+C,1)) BITAND 7Fh)
        IF C$ > "<1Fh>" THEN
          RL$ = RL$ + C$
        ELSE
          RL$ = RL$ + "."               ' einen Punkt fuer nicht druckbare Zeichen
        ENDIF
      NEXT
      PRINT #SER,#1, "  ";RL$           ' ASCII-Zeichen und neue Zeile
    NEXT
    PRINT #SER,#1, "<13><10>";          ' leere Zeile
  NEXT
  PRINT #SER,#1, "<13><10>fertig"       ' "fertig" drucken, wenn alles angezeigt
END                                     ' wurde


'-------------------------------------------------------------------------------
' Unterprogramm EE_RD zum EEPROM-Lesen
'-------------------------------------------------------------------------------
SUB EE_RD                               ' EEPROM lesen (8Seiten)

  LONG RFLAG, P, Q
  STRING RP$(256), RQ$(32)              ' maximale Anzahl Bytes: 32

  FOR P = 0 TO N_PAGES-1                ' nacheinander Seiten 0-7
    EE_ADR = M24LC16 BITOR 2*P          ' Seiten-Adresse
    RP$ = ""
    FOR Q = 0 TO 7                      ' Lesen als 32 Byte snaps
'    A$ =   I2C_READ$   (i2c_sladr, i2c_adr, nob) ' Lese Byte(s)
      RQ$ = I2C_READS$  (   EE_ADR,    Q*32,  32) ' Lese (Limit 32 bytes)
      RFLAG = I2C_RESULT()              ' Anzeige des Lese-Status
      PRINT USING #SER, #1, "<13>Rd=";RFLAG;" page ";P;" Q=";Q;" ";
      RP$ = RP$ + RQ$                   ' sammle snaps
    NEXT                                ' naechster snap
    R$(P) = RP$                         ' sammle Seite
  NEXT                                  ' naechste Seite
  PRINT #SER,#1, "<13><10>";            ' neue Zeile
END                                     ' Ende Unterprogramm
'-------------------------------------------------------------------------------


'-------------------------------------------------------------------------------
```

```
' Name: EEPROM_S.TIG
' Eine Chipkarte mit EEPROM M24LC16 (2kByte) wird ueber den I2C-Bus mit
' verschiedenen Daten seitenweise beschrieben. Jede Seite hat 256 Bytes.
' Mit dem Programm EEPROM_L kann der Inhalt als Hex-Dump auf einen PC ueber-
' tragen.
'
' Anbindung BASIC-Tiger -> Chipkarte (oder EEPROM)
' Vcc  -> Vcc
' Vss  -> Vss
' L70 BASIC-Tiger -> Data
' L72 BASIC-Tiger -> Clock
' L71 Kathode Diode -> Anode Diode -> Data
'
' Einstellungen des Terminalprogrammes
' 19200 Bd, keine Paritaet, Xon/Xoff, 1 Stopbit
'-------------------------------------------------------------------------------
user_var_strict                         ' strenge Variablen-Deklaration

#include DEFINE_A.INC                   ' Include-Dateien
#include UFUNC3.INC                     ' einbinden

#define M24LC16 0A0h                    ' Adresse des EEPROM Seite 0
#define N_PAGES 8                       ' Anzahl der Seiten = 8
ARRAY W$(8) OF STRINGS(256)             ' Array fuer die zu schreibenden Strings
BYTE EE_ADR                             ' Variable fuer die Seiten-Adresse

'-------------------------------------------------------------------------------
TASK MAIN                               ' Beginn der Task Main

  LONG C, I, L, P, Q                    ' Variablen-Deklaration
  BYTE SP, A
  STRING C$(1), RA$(256), RL$(16)
  STRING H1$, H2$(256)


'-------------------------------------------------------------------------------
' Vorbelegung des String-Arrays W$(x) mit 8 Strings (8 Seiten)
'-------------------------------------------------------------------------------
W$(0) = "Seite 0                          &  ' Seite 0 mit normalem
Text                            &           ' Text beschreiben
Das ist ein Bei-spiel fuer nor- &           ' (W$(x) vorbelegen)
malen Text im   EEPROM. Der Text&
kann als norma- ler String ein- &
geben werden.                   &
                              &
                                &
                                "


H1$ = "Seite 1                          &    ' Seite 1 mit Hex-Daten
Hex-Daten                       "           ' beschreiben (W$(x) vorbelegen)
H2$ ="00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F &
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F &
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F &
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F &
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F &
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F &
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F &
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F &
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F &
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F &
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF &
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF "%
W$(1) =H1$+H2$

W$(2) = "Seite 2                          &  ' Seite 2 mit 0
Seite auf 0 set-zen             "           ' beschreiben
  FOR C = 32 TO 255                         ' (W$(x) vorbelegen)
    W$(2) = W$(2) + CHR$(0)
  NEXT
```

```
W$(3) = "Seite 3                          &  ' Seite 3 mit FF
Seite auf FF    setzen          "            ' beschreiben
  FOR C = 32 TO 255                          ' (W$(x) vorbelegen)
    W$(3) = W$(3) + CHR$(255)
  NEXT

W$(4) = "Seite 4                          &  ' Seite 4 mit einer
Beispiel fuer Datenbank         &            ' "Datenbank" beschreiben
Name:           &                            ' so kann man z.B. ein
  Hecht         &                            ' Zutrittssystem einschl.
Vorname:        &                            ' Passwort anlegen
  Hubert        &                            ' (W$(x) vorbelegen)
Wohnort:        &
  Karpfenteich  &
Telefon:        &
  0123456789    &
Passwort:       &
  James Bond 007&
Beruf:          &
  Hacker        "

W$(5) = "Seite 5                          &  ' Seite 5 mit 0
Seite auf 0     setzen          "            ' beschreiben
  FOR C = 32 TO 255                          ' (W$(x) vorbelegen)
    W$(5) = W$(5) + CHR$(0)
  NEXT

W$(6) = "Seite 6                          &  ' Seite 6 mit 0
Seite auf 0     setzen          "            ' beschreiben
  FOR C = 32 TO 255                          ' (W$(x) vorbelegen)
    W$(6) = W$(6) + CHR$(0)
  NEXT

W$(7) = "Seite 7                          &  ' Seite 7 mit 0
Seite auf 0     setzen          "            ' beschreiben
  FOR C = 32 TO 255                          ' (W$(x) vorbelegen)
    W$(7) = W$(7) + CHR$(0)
  NEXT
'-------------------------------------------------------------------------------

' I2C_SETUP  (pin, din, pout, dout, clout, spcnt)
  I2C_SETUP  (  7,   0,    7,    1,     2,     0)

  CALL EE_WR                              ' 8 Seiten schreiben
END

'-------------------------------------------------------------------------------
' Unterprogramm EE_WR zum EEPROM-Schreiben
'-------------------------------------------------------------------------------
SUB EE_WR                                 ' in EEPROM schreiben
  LONG P, Q, WP, WFLAG, X
  STRING WP$(8)                           ' snap fuer EEPROM

  PRINT #SER, #1, "writing..."
  FOR P = 0 TO N_PAGES-1                  ' Seiten
    FOR Q = 0 TO 31                       ' 32 snaps zu je 8
      WP$ = MID$ ( W$(P), Q*8, 8 )
      WP = LEN ( WP$ )
      X = 0                               ' Fuer I2C_ACK_POLL
      EE_ADR = M24LC16 BITOR 2*P          ' Ausrechnen der Seiten-Adresse
'      I2C_WRITE (i2c_sladr, i2c_adr, nob,   var$) ' Schreiben
      I2C_WRITE (    EE_ADR,     Q*8,  WP,    WP$) ' 1 Byte
                                          ' poll EEPROM busy
      IF I2C_ACK_POLL ( EE_ADR ) = 1 THEN
        X = X + 1                         ' poll counter
      ENDIF            ' Bemerkung: das Polling ist nicht wirklich notwendig
      WFLAG = I2C_RESULT()                ' abfragen des Schreib-Status
                                          ' kann benutzt werden, um
                                          ' Error-Behandlung auszuloesen
    NEXT
```

```
  NEXT
END                                    ' Ende Unterprogramm
'-------------------------------------------------------------------------------
```

Viel Spaß beim Experimentieren mit den I^2C-Chipkarten. Die Hardware sowie die vorgestellten Routinen ermöglichen auch den Einstieg in die Technik anderer Chipkarten.