Chipkarten

Ich kenne noch die Diskussion um die Einführung der Scheckkarte und später auch der Telefonkarten. Es wurde uns immer wieder erzählt, dass diese Karten absolut fälschungssicher seien, da der technische Aufwand zum Knacken überaus hoch sei. Dies galt sicherlich noch in den 70er Jahren für die Scheckkarte, da man zum Lesen der Information einen Computer und ein Lesegerät für die Magnetinformation auf der Rückseite der Karte besitzen mußte. Da waren noch einige tausend Mark dazwischen.

Doch schon Anfang der 80er Jahre war die Möglichkeit des Auslesens und des Beschreibens einer Magnetspur keine Hexerei mehr. Da bei uns alles schön genormt ist, konnte man sich auch die Spezifikationen der ISO – Norm besorgen. Hierin war dann fein säuberlich aufgezeigt, wo was in welcher Spur und wieviel Bytes lang usw. zu stehen hat.

Deshalb plaudere ich hier keine Geheimnisse aus und werde auch keine Anleitungen zum Missbrauch geben.

1985 gestalteten wir eine Sendung über die Möglichkeiten der Manipulation von Scheckkarten. Diese Sendung schlug wie eine Bombe ein und wir bekommen heute noch Post mit der Bitte quasi die Rolle eines Gutachters einzunehmen und zu bestätigen, dass es möglich ist, die Pinnummer aus dem Inhalt auf der Karte zu erspähen. Dieses haben wir allerdings nie behauptet. Damals ging es darum, zu zeigen, dass man während eines Tages mehrmals das Limit (damals 400.-DM) aus einem Bankautomaten zu holen.

Die Entwickler hatten das Problem ganz toll gelöst: hob man heute Geld aus einem Bankautomaten, so wurde und wird das Datum auf eine Spur des Magnetstreifens zurückgeschrieben. Geht man am selben Tag zu diesem Automaten oder zu einem anderen Automaten, so liest dieser zuerst von der Chipkarte das Datum und entscheidet: wird das Datum von heute gelesen, so wird eine zweite Geldausgabe verweigert.

Was aber ist, wenn ich nach dem ersten Abheben heute die Magnetspur manipuliere und ein anderes Datum aufbringe? Zunächst einmal ist dies natürlich kriminell – zum anderen spuckte damals fast jeder Automat die 400 Mark erneut aus.

Das Wort –damals- ist wichtig. Damals waren die Bankautomaten meistens Standalone - Geräte, die keine Verbindung zur Zentrale hatten, sie wurden offline betrieben. Daher ist es klar, dass der Geldautomat B keine Ahnung hatte, dass am Automaten A heute bereits abgehoben wurde.
Erst nach und nach wurden die Automaten vernetzt. Damit wußte der Zentralcomputer natürlich um jeden Abhebevorgang.

Zum Teil aber waren sie am Wochenende noch von der Zentrale abgenabelt. Es dauerte lange –und ist wahrscheinlich noch nicht komplett abgeschlossen- bis alle Auslandsautomaten Zugriff auf die Daten online bekamen. DFÜ war lange Zeit ein Spezialgebiet und voller Rätsel.

Die Faszination, die solche Speicherelemente wie Scheckkarten, Telefonkarten, Krankenkarten, Chipkarten jeder Couleur ausmachen, ist für einen eingefleischten Techniker sicherlich gegeben. Was steckt hinter dieser Technik und wie kann ich evtl. die Inhalte auslesen?

Wieviel Guthaben ist auf meiner Telefonkarte?

Die Beantwortung dieser Frage dürfte legitim sein, zumal es genügend preiswerte Lesegeräte für das Auslesen von Telefonkarten gibt. Die Idee, die CControl in Verbindung mit einem Kartenadapter als wirtschaftliche Idee darzustellen, dürfte daher nicht ganz ernst genommen werden. Das wäre viel zu teuer.

Doch ich hoffe den Spieltrieb wecken zu können, um mit eigenen Experimenten technische Vorgänge besser verstehen zu können. Erst dann kommt man vielleicht zu der Erkenntnis, dass alle „nur mit Wasser“ kochen und sich das „Komplizierteste“ in eine simple Abfolge von Bits und Bytes darstellt.

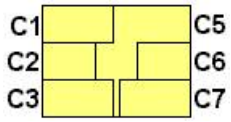
Wie kommen wir an das Restguthaben ?

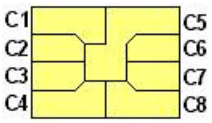
Wie immer muß man etwas über Normen wissen, denn diese Speicherkarten müssen ja auch von allen Automaten wieder gelesen werden können, so dass wahrlich nicht jeder sein Süppchen kochen darf. Den wichtigsten Standard und damit die Grundlage für die kontaktbehafteten Chipkarten "Integrated circuit(s) cards with contacts" bildet die ISO - Norm 7816. Die Norm wird ständig erneuert.
Hier ist alles definiert, was zur reibungslosen Funktion benötigt wird: die physikalischen Eigenschaften einer Chipkarte, die Abmessungen und die Lage der Kontakte, die elektronischen Signale und die Kommunikationsprotokolle usw. Für die Realisierung der gestellten Aufgabe brauchen wir wahrlich nicht alles zu wissen.

Das Wichtigste ist die Kontaktbelegung. Sie wird praktisch von allen Chipkarten verwendet, bis auf Ausnahmen – wie immer. Es gibt neben der ISO – Variante noch eine sog. AFNOR – Variante, bei der die Belegung praktisch um 180 ° gedreht ist. Noch nie kam mir allerdings eine AFNOR – Karte in die Hände, so dass man sich sicherlich auf die ISO-Variante beschränken kann.
Man unterscheidet noch zwischen 6-poligen und 8-poligen Chips. Die 6-polige Version ist dabei kompatibel zu der 8-poligen Version, es wurden hier lediglich die zwei unteren, unbenutzten Pole eingespart.

Wenn man sich verschiedene Telefonkarten anschaut, so entdeckt man ein unterschiedliches Layout der Anordnung der Kontakte. Technisch macht dieses keinen Unterschied. Dem Kenner genügt ein Blick auf die Oberfläche und er sagt: „Aha, von dem und dem Hersteller..“.

Zwei verschiedene Chiplayouts sind unten abgebildet. Die erste Version ist sechspolig, die zweite achtpolig. Für unsere Zwecke ist dies gleichbedeutend. C4 und C8 werden nicht benutzt.





Die Funktion der einzelnen Kontakte ist in der Tabelle aufgezeigt.

C1 = Vcc (+5 V)	C2 = Reset (rst)	C3 = Clock (clk)	C4 = XXX
C5 = Masse	C6 = XXX	C7 = I/O (dat)	C8 = XXX

Das Wichtigste wäre damit geschafft. Wer ein bisschen Erfahrung mit Datenübertragung hat, der kann schon erkennen, dass wahrscheinlich eine synchrone Übetragung stattfinden wird. Zunächst einmal wird man einen Reset über C2 ausführen, dann werden 8 Clockimpulse clk gegeben. Zwischen jedem Impuls kann das Ergebnis über den I/O-Pin dat eingelesen werden.

Der Kopf des Programmes sieht daher so aus :
(je nach eigener Festlegung der Ports)

```
define dat      port[14]
define rst      port[15]
```

```
define clk      port[16]
```

Zunächst einmal interessiert uns lediglich das Restguthaben auf der Telefonkarte. Dazu muß man wieder das Format kennen, wie die Beträge abgespeichert sind. Es ist irgendwie seltsam aufgebaut.

Die ersten 8 Bytes (Speicherzellen 0,1,2,3,4,5,6,7) beinhalten zusätzliche Informationen und Kennungen zur Karte und sind schreibgeschützt.

Die folgenden 5 Bytes sind beschreibbar und natürlich auch lesbar. Hier stehen die Bits für das Restguthaben. Die Beträge werden nicht nach der Wertigkeit 1,2,4,8,16,32 usw. berechnet, sondern nach einem 5-stufigen Oktalzähler. Wir werden gleich sehen, was dieses bedeutet.

Jedes einzelne Byte enthält eine Anzahl auf 1 gesetzter Bits. Diese 1-en werden gezählt. Das Ergebnis ist dann ein Wert zwischen 0 und 8.

Beispiel 1: Byte 00000101 → Wert = 2
Beispiel 2: Byte 00001001 → Wert = 2
Beispiel 3: Byte 10000100 → Wert = 2
Beispiel 4: Byte 10110101 → Wert = 5

Die 5 Bytes haben eine bestimmte Wertigkeit, die sich nach der Basis 8 (oktal) richtet.

Byte 1 (Speicherzelle 8) = 8 * 8 * 8 * 8 = 8^4 = 4096
Byte 2 (Speicherzelle 9) = 8 * 8 * 8 = 8^3 = 512
Byte 3 (Speicherzelle 10) = 8 * 8 = 8^2 = 64
Byte 4 (Speicherzelle 11) = 8 = 8^1 = 8
Byte 4 (Speicherzelle 12) = 8^0 = 1

Das Restguten ermittelt sich nach der Anzahl der gezählten Bits in einer Speicherstelle mal der Wertigkeit der Speicherzelle.

Beispiel:

Speicherzelle 8 (Wertigkeit 4096)	: an einsen gezählt: 1 =	4096
Speicherzelle 9 (Wertigkeit 512)	: an einsen gezählt: 2 =	1024
Speicherzelle 10 (Wertigkeit 64)	: an einsen gezählt: 5 =	320
Speicherzelle 11 (Wertigkeit 8)	: an einsen gezählt: 4 =	32
Speicherzelle 12 (Wertigkeit 1)	: an einsen gezählt: 8 =	8
		———
		5480 Einheiten

Für eine deutsche Telefonkarte also 5480 Pfennige oder 54,80 DM Restguthaben!

Jetzt ist das Programm zum Auslesen des Restbetrages sicherlich kein Problem mehr. In fortführung an den oben bereits aufgezeigten Definitionskopf können wir jetzt die weiteren Variablen definieren.

```
define dat      port[14] ' I/O - Port
define rst      port[15] ' Resetport
define clk      port[16] ' Clockport

define wert     word     ' Ergebnis (word)
define bits     byte     ' Anzahl Bits
define n        byte     ' Schleifenzähler
define i        byte     ' Schleifenzähler
deact dat                ' dat als Eingang
```

```
#init
rst=1              ' Reset high
for n=0 to 15      ' einige mal
pulse clk          ' clocken
next n             '
rst = 0            ' Reset low

#dummyread
for i=0 to 7       ' die ersten 8
gosub bitread      ' Bytes einlesen
next i             ' (ohne Verwertung)

wert = 0
gosub bitread
wert = wert + (4096 * bits)
gosub bitread
wert = wert + (512 * bits)
gosub bitread
wert = wert + (64 * bits)
gosub bitread
wert = wert + (8 * bits)
gosub bitread
wert = wert + (1 * bits)

print "Restguthaben: "; wert /100 ; ".";wert mod 100; " DM"

end

#bitread
bits = 0
for n=1 to 8
if dat = on then bits = bits + 1
pulse clk
next n
return
```

Dieses kleine Programm müsste schon das Gewünschte anzeigen: Soundsoviele DM Restguthaben auf der Karte.

Was aber versteckt sich in den ersten 8 Bytes, die im obigen Beispiel ja übersprungen wurden? Ehrlich gesagt, so ganz genau weiß ich das auch nicht. Ich habe hier unterschiedliche Interpretationen gelesen und überall fehlt die eine oder andere Information. Offensichtlich ist die ISO – Norm nur für teures Geld zu kaufen und wer sie besitzt, gibt nicht unbedingt das Herrenwissen preis.

Es ist aber kein Problem, diese ersten Bytes darzustellen, um an dem Inhalt zu deuteln. Das Programm dazu ist noch einfacher als vorhin. Dargestellt werden die ersten Bytes in binärer Form und zwar als Doppelnibbel. Nibbels sind 4 Bit breite Informationen und können Werte zwischen 0 und 15 darstellen. Der linke Teil ist dann das Highnibbel, entsprechend der rechte Teil das Lownibble.

```
define dat      port[14] ' I/O - Port
define rst      port[15] ' Resetport
define clk      port[16] ' Clockport
```

```
   define wert     word     ' Ergebnis (word)
   define bits     byte     ' Anzahl Bits
   define n        byte     ' Schleifenzähler
   define i        byte     ' Schleifenzähler

   deact dat          ' dat als Eingang

   #init
   rst=1              ' Reset high
   for n=0 to 15      ' einige mal
   pulse clk          ' clocken
   next n             '
   rst = 0            ' Reset low


   for i=0 to 7       ' die ersten 8 Bytes
   print "Byte ";i; "  ";
   gosub byteread
   print
   next i

   end

   #byteread
   bits = 0
   for n=1 to 8
   if dat = on then print "1"; else print "0";
   if n=4 then print " ";
   pulse clk
   next n
return
```

So sieht zum Beispiel eine Telefonkarte im Ergebnis aus, die die folgende Aufschrift trägt:
ODS A 08 03.94 80.000.
ODS steht für `Oldenbourg Datensysteme (ODS)`

Ein zweites Beispiel rechts. Hier lautet die Aufschrift
O 574 04.94 3.000 DTMe