Matthias Heuschele

Chipkartenprogrammierung mit der Basic-Card

2014

1.0 Grundlagen

1.1 Was ist eine Chipkarte

In jeder Geldbörse oder Brieftasche finden sich heutzutage ein oder mehrere Plastikkärtchen, die umgangssprachlich einfach als Chipkarten bezeichnet werden. Was jedoch ist eine Chipkarte bzw. wie definiert man den Begriff Chipkarte. Grundsätzlich spricht man von einer Chipkarte, wenn auf dieser ein so genanntes Chipmodul integriert ist, dessen goldenes Erscheinungsbild Ihnen sicher schon in der ein- oder anderen Form begegnet ist.

Abbildung 1

Prinzipiell gibt es zweierlei Typen von Chipkarten, die Speicherkarte und die Mikrocontrollerkarte. Auf einer Speicherkarte werden Informationen nur gespeichert und bei Bedarf wieder ausgelesen. Dieser Speicher ist im Regelfall gegen nicht autorisierte Zugriffe speziell abgesichert und der Speicherinhalt wird je nach Speicherchip teilweise intern verschlüsselt.
Eine Mikrocontrollerkarte beinhaltet dagegen –so unglaublich es klingen mag- einen vollwertigen Computer nebst Speicher, Prozessor, sowie Eingabe- und Ausgabeschnittstellen. Mit dieser Art von Chipkarten wollen wir uns in diesem Buch beschäftigen, indem wir diesen „Computer“ programmieren.
Der Vollständigkeit halber möchte ich noch die kontaktlose Chipkarte vorstellen, die besser unter dem Begriff RFID bekannt ist. Bei einer RFID Karte handelt es sich auch um eine Chipkarte, allerdings erfolgt die Kommunikation mit der Außenwelt nicht über eine fest verdrahtete Verbindung, sonder über Funk.

1.2 Aufbau einer Mikrocontrollerkarte

1.2.1 Die Kontaktfläche

Wenn Sie noch mal Abbildung 1 betrachten, werden Sie feststellen, dass dieses Bild nicht im Entferntesten einem Computer ähnlich sieht. Was Sie hier sehen sind allerdings nur die Kontakte des Chipmoduls, das den darunter liegenden Mikrocontroller mit der Außenwelt verbindet und mit Strom versorgt. Die Kontaktfläche ist in 6 oder 8 einzelne Bereiche unterteilt, die teilweise abenteuerliche Formen aufweisen können. Die Positionen, Größen und Abstände der einzelnen Kontaktbereiche sind dagegen in einer Norm festgelegt, womit sichergestellt wird, dass jedes Kartenlesegerät auch den richtigen Kontakt „trifft". Mikrocontrollerkarten mit nur 6 Bereichen bzw. Kontakten bieten dieselbe Funktionalität wie Mikrocontrollerkarten mit 8 Kontakten. Dieses Resultiert daraus, dass bei Mikrocontrollerkarten mit 8 Kontakten nicht alle Kontakte belegt, bzw. für spätere Anwendungen reserviert sind.

Schauen wir uns nun die Belegung der einzelnen Kontakte an, die natürlich auch normiert sind. Jede Kontaktfläche hat eine Nummer, der ein C für Contact vorangestellt wird. Die Nummerierung erfolgt fortlaufend und beginnt mit dem Kontakt links oben:

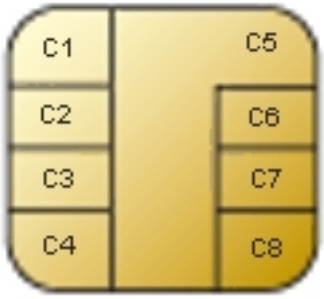

Abbildung 2

Kontakt	Belegung	Bedeutung
C1	Vcc	Versorgungsspannung
C2	RST	Reset
C3	CLK	Takt
C4	AUX1	Reserviert
C5	GND	Masse
C6	Vpp	Programmierspannung - Wird nicht mehr verwendet.
C7	I/O	Serielle Ein- und Ausgabe
C8	AUX2	Reserviert

Tabelle 1

C1 – Vcc - Versorgungsspannung
Damit der „Computer" der Prozessorkarte überhaupt funktioniert, muss dieser natürlich mit einer Spannung versorgt werde. Heutige Mikrocontrollerkarten werden in der Regel mit 3 oder 5 Volt betrieben. Manche Mikrocontrollerkarten können wahlweise sogar mit beiden Spannungen betrieben werden.

C2 – RST - Reset
Über diesen Kontakt wird die Mikrocontrollerkarte zurückgesetzt. Die Funktionsweise ist vergleichbar mit einem Hardware Reset am PC, bei dessen Ausführung der PC neu gestartet wird.

C3 – CLK - Takt
Wie fast jeder Prozessor benötigt auch die Mikrocontrollerkarte eine externe Taktversorgung. Der Takt, der über den Kontakt C3 zur Karte gelangt, wird bei den meisten Mikrocontrollerkarten nicht direkt dem

Prozessor zugefügt, sondern intern, entsprechen dem Verwendungszweck, vervielfacht oder geteilt. So wird z.B. der Prozessor auf der Mikrocontrollerkarte üblicherweise mit einem vielfachen der angelegten Frequenz getaktet, während andere Teile des Mikrocontrollers vielleicht nur die halbe Taktfrequenz benötigen.

C4 –AUX1 - Reserviert
Der Kontakt ist für zukünftige Anwendungen reserviert und wird zurzeit nur von den wenigsten Mikrocontrollerkarten benutzt.

C5 –GND - Masse
Der Minuspol halt.

C6 –Vpp - Programmierspannung
Über diesen Kontakt wurden die Eeproms früherer Mikrocontrollerkarten mit einer so genannten Programmierspannung versorgt und somit Daten in das Eeprom geschrieben oder auch gelöscht. Heutzutage werden diese Programmierspannungen von der Mikrocontrollerkarte selbst erzeugt, so dass dieser Kontakt nun abkömmlich geworden ist.

C7 –I/O - Serielle Ein- und Ausgabe
Der Verwendungszweck einer Mikrocontrollerkarte wäre äußerst sinnlos, wenn diese nicht mit der Außenwelt kommunizieren könnte. Da heutige Mikrocontroller noch über keine grafische Benutzeroberfläche verfügen (ein kleiner Scherz), kommuniziert eine Mikrocontrollerkarte über elektrische Signale mit der Außenwelt. Da zur Datenübertragung nur der C7 Kontakt zur Verfügung steht, erfolgt die Datenübertragung im Halbduplexverfahren, bei dem entweder gesendet ODER empfangen werden kann. Das Übertragungsprotokoll ist sehr eng an das seriellen Protokoll der RS232 Schnittstelle angelegt, weshalb jede Mikrocontrollerkarte über einen UART-Baustein verfügt, der auch –in abgeänderter Form- auf jedem PC Mainboard zu finden ist.

C8 –AUX2 - Reserviert
Der Kontakt ist für zukünftige Anwendungen reserviert und wird zurzeit nur von den wenigsten Mikrocontrollerkarten benutzt.

1.2.2 Der Mikrocontroller

Unter der Kontaktfläche befindet sich er eigentliche Mikrocontroller der Chipkarte. Wenn ich im Folgenden von Chipkarte spreche, meine ich damit immer eine Mikrocontrollerkarte und keine Speicherkarte.
Wie in Kapitel 1.1 bereits erwähnt, ist ein Mikrocontroller ein vollwertiger Computer mit Prozessor, Speicher und I/O Schnittstellen. Da Chipkarten hauptsächlich in sicherheitsrelevanten Bereichen eingesetzt werden, haben Mikrocontroller für Chipkarten, im Gegensatz zu herkömmlichen Mikrocontrollern, zusätzliche Baugruppen integriert, deren Verwendungszweck einzig darin besteht, die auf der Chipkarte vorhandenen Programme und Daten vor unbefugten Zugriffen zu schützen. Diese Zusatzhardware kann beispielsweise aus einzelnen Detektoren (Licht, Takt, Frequenz, Temperatur, Spannungs- und Stromdetektoren, etc.) bestehen, die auf äußere Manipulationen der Chipkartenhardware entsprechend reagieren.
Oftmals besteht der Anwendungszweck einer Chipkarte darin, bestimmte Daten in relativ kurzer Zeit zu verschlüsseln bzw. zu entschlüsseln. Aus diesem Grund enthalten viele Chipkarten zusätzlich so genannte Kryptoprozessoren, deren Hardware diese Aufgabe in viel kürzerer Zeit erledigen kann, als eine reine Softwarebasierte Lösung.

Bevor ich Ihnen die gebräuchlichsten Bestandteile eines Chipkarten-Mikrocontrollers vorstelle, hier eine kurze Auflistung oft verwendeter Mikrocontrollertypen und deren Kennwerte:

Hersteller	Bezeichnung	CPU	ROM	Flash	EEPROM	RAM	Takt (max.)
Atmel	AT90SC3232C	AVR	-	32 KB	32 KB	3 KB	16 MHz
Atmel	AT90SC6464C	AVR	-	64 KB	64 KB	3 KB	16 MHz
Atmel	AT90SC7272C	AVR	-	72 kB	72 KB	3 KB	16 MHz
Atmel	AT90S8515-8	AVR	-	8 KB	0,5 – 1024 KB	0,5 KB	8 MHz
ST Microelectronics	ST16SF42	6805	16 kB	-	2 KB	384 Byte	5 MHz
Microchip	PIC16F84A	PIC	-	1 KB	64 Byte	68 Byte	20 MHz
Hitachi	AE460	AE-4	96 KB	-	64 KB	4,5 KB	10 MHZ
Infineon	SLE66C160P	8051	64 KB	-	16 KB	2 KB	10 MHZ
Philips	P8WE6004	8051	32 KB	-	4 KB	768 Byte	8 MHz

Tabelle 2

Prozessor

Oftmals wird irrtümlich angenommen, dass Chipkartenprozessoren Spezialentwicklungen sind, die eigens für diesen Einsatz entworfen wurden. Diese Annahme ist falsch, denn ein Prozessorhersteller entwickelt seine Prozessoren in allen möglichen Ausführungen (DIL, SMD), unter anderem auch in einer Chipkartenausführung. In den meisten Fällen handelt es sich aber gar nicht mehr um einen reinen Prozessor, sondern gleich um einen ganzen Mikrocontroller, in dem bereits verschiedene Peripheriefunktionen integriert sind. Die Abgrenzung vom reinen Prozessor zum Mikrocontroller ist somit etwas fließend.
Es ist aber dennoch wichtig zu wissen, welcher Prozessor auf dem Mikrocontroller Verwendung findet, denn jeder Prozessortyp hat seinen eigenen speziellen Befehlssatz, mit dem dieser Prozessor programmiert werden kann. Auf Hochsprachenebene (C, Basic, Java, usw.) bedeutet dies, dass der Compiler bzw. Interpreter die Hochsprachenbefehle in so genannte Maschinenbefehle umwandeln muss, die genau dieser Prozessor „versteht". Somit wird auch verständlich, dass ein Programm, das für einen bestimmten Prozessortyp geschrieben wurde, nicht nur auf der Chipkarte lauffähig ist, sondern auf allen Systemen, die auf diesem Prozessortyp basieren. Allerdings gilt dies natürlich nur rein auf Prozessorebene, denn die Peripherie kann bei den einzelnen Systemen erheblich voneinander abweichen.

Speicher

Auf Chipkarten werden hauptsächlich 4 verschiedene Speichertypen eingesetzt: ROM, RAM, EEPROM und Flash-EEPROM.

ROM (Read Only Memory)

Wie der Name „Read Only Memory" schon aussagt, kann aus diesem Speicher nur gelesen werden. In der Regel wird das ROM bereits vom Chipkartenhersteller mit dem Betriebssystem der Chipkarte, oder mit diversen Diagnosefunktionen beschrieben. Für Sie als angehender Chipkartenprogrammierer hat dieser Speicher keine Bedeutung.

RAM (Random Access Memory)

Das RAM ist ein so genannter flüchtiger Speicher, der nur verwendet werden kann, solange die Chipkarte in Betrieb ist, d.h. solange diese mit Strom versorgt wird. Mit diesem Speichertyp werden Sie noch oft konfrontiert werden, denn im RAM werden all Ihre Programmvariablen, Strings und temporäre Daten gespeichert. Allerdings werden diese Daten nach dem ausschalten der Chipkarte wieder gelöscht, da diese Speicherart ja „flüchtig" ist. Als Programmspeicher ist dieser Speichertyp somit nicht geeignet, denn Ihre Programme und Daten sollen ja nach dem ausschalten der Chipkarte weiterhin auf dieser verbleiben.

EEPROM (Electrical Erasable Read Only Memory)

Das EEPROM ist ein nichtflüchtiger Speicher, der aber im Gegensatz zum ROM gelöscht und wieder beschrieben werden kann. Allerdings kann dies nicht unbegrenzt erfolgen, da das EEPROM im Prinzip aus Kondensatoren besteht, deren Lebensdauer leider beschränkt ist. Heutige EEPROMs können bis zu 1 000 000 mal wieder beschrieben werden, was sicher eine ganze Zeit ausreicht.
Sie können sich das EEPROM als eine Art Festplatte vorstellen, auf der Sie als Chipkartenprogrammierer Ihre Programme und permanente Daten ablegen können.

Flash-EEPROM

Das Flash-EEPROM, das allgemein nur Flash genannt wird, ist im Prinzip ein sehr schneller EEPROM-Speicher, auf den etwa 1000-mal schneller zugegriffen werden kann, als auf ein herkömmliches EEPROM.

I/O Bausteine

Damit eine Chipkarte mit der Außenwelt – die in diesem Fall aus einem Terminal (Chipkartenlese-Schreibgerät) besteht – kommunizieren kann, ist im Mikrocontroller ein so genannter UART-Baustein (Universal Asynchronous Receiver) integriert, der die serielle Datenübertragung von und zur Chipkarte ermöglicht. Die Datenübertragung erfolgt prinzipiell nach dem RS232-Protokoll, allerdings in einer abgespeckten Variante, da ja nur eine einzige Leitung (Kontakt C7) zur Verfügung steht.
Spezielle Leitungen wie RTS, CTS, DSR, die bei der RS232-Schnittstelle Verwendung finden, stehen bei der Kommunikation mit der Chipkarte nicht zur Verfügung oder werden auf Softwarebasis nachgebildet. Die Datenübertragung erfolgt wie bereits erwähnt, im Halbduplexverfahren. Als Übertragungsraten werden die RS232 üblichen Bitraten wie 1200 bps, 4800 bps, 9600 bps usw. verwendet. Ebenso enthalten die übertragenen Daten die RS232 üblichen Synchronisierungsdaten wie Startbits, Stoppbits und Paritätsbits.
Der Vollständigkeit halber sollte ich noch erwähnen, dass in einigen Chipkartenmikrocontrollern bereits USB-Schnittstellenbausteine integriert sind, wodurch die Chipkarte über spezielle USB-Terminals angesprochen werden kann. Da die Verwendung der USB-Schnittstelle im Bereich Chipkarten noch nicht genormt ist, wird die Kontaktbelegung allein vom Chipkartenhersteller festgelegt. Um Inkompatibilitäten zu umgehen, haben sich einige Chipkartenhersteller darauf geeinigt, die bisher unbenutzten Kontakte C4 und C8 für die Kommunikation per USB zu verwenden.

Zusatzhardware

In der Regel enthalten Mikrocontroller für Chipkarten einige Bauelemente, die auf herkömmlichen Mikrocontrollern nicht zu finden sind. Diese Bauelemente dienen einerseits zur Sicherung der Kartendaten vor unbefugten Zugriffen, als auch zur Berechnung mathematischer Funktionen wie Prüfsummen, Zufallszahlen, Verschlüsselung, usw., deren Berechnung auf Softwareebene zu lange dauern würde.
Die Bauteile die eine Chipkarte vor physikalischen Angriffen schützen sollen, bestehen meist aus Detektoren wie Licht, Takt, Frequenz, Temperatur, Spannungs- und Stromdetektoren, etc. So ist z.B. der Baustein für die Taktüberwachung dafür zuständig, dass der Mikrocontroller nicht außerhalb Taktspezifikation liegt. Würde man den Takt nicht überwachen, könnte ein Angreifer den Mikrocontroller derart Untertakten, dass der Prozessor quasi im Einzelschrittmodus (Debuging-Modus) betrieben wird. Dadurch könnte der Angreifer nun in aller Ruhe die Spannungen und Ströme messen, die der Prozessor gerade aufnimmt. Mit diesen Werten könnte über eine Referenztabelle ermittelt werden, welchen Befehl der Prozessor gerade ausführt. Wenn man diesen Vorgang längere Zeit fortsetzen würde, könnte man das komplette Programm der Chipkarte auslesen.
Nun ist es aber so, dass erfolgreiche physikalische Angriffe eher selten sind, denn dieser Angriffstyp setzt eine umfangreiche Ausstattung, ein entsprechendes Budget und natürlich umfassendes Know-how voraus. Vielmehr enthüllt eine Chipkarte Ihre Geheimnisse weit öfters Aufgrund fehlerhafter Softwareimplementierungen. Und da diese Angelegenheit Sie ja nun ganz im Besonderen Betrifft, habe ich diesem Thema ein eigenes Kapitel gewidmet.

1.3 Chipkarten Terminals

Wenn umgangssprachlich von einem Smartcard-Reader oder Chipkarten-Lesegerät die Rede ist, weis der „Fachmann“, dass damit ein Chipkarten-Terminal gemeint ist. In diesem Abschnitt möchte ich den Begriff Chipkarten-Terminal etwas genauer definieren, und Ihnen erläutern welche Funktionen das Terminal übernimmt.

Wie bereits gesagt, wird ein Chipkarten-Terminal umgangssprachlich als Smartcard-Reader –auf Deutsch Chipkarten-Lesegerät- bezeichnet. Aus dieser Namensgebung ergibt sich der Eindruck, dass es sich dabei um ein Gerät handelt, mit dem Chipkarten nur gelesen werden können. Wenn man diesen Gedanken weiterverfolgt, könnte man daraus schließen, dass zum Beschreiben einer Chipkarte zusätzlich noch ein „Smartcard-Writer“ benötigt wird.
Wie Sie sicher noch wissen, werden Daten von und zu einer Chipkarte über den Kontakt C7 übertragen. Wenn Sie also Daten von einer Chipkarte lesen möchten, werden diese Daten seriell über den Kontakt C7 an den Smartcard-Reader gesendet. Möchten Sie dagegen Daten auf die Chipkarte schreiben, werden die Daten vom Smartcard-Reader wiederum über den Kontakt C7 zu Chipkarte gesendet. Daraus wird ersichtlich, dass ein Smartcard-Reader und eine Chipkarte einfach nur zwei Geräte sind, die Daten miteinander austauschen, genau wie zwei PCs die per R2232-Schnittstelle oder LAN-Kabel miteinander verbunden sind. Da Sie und ich nun wissen, dass der Begriff Smartcard-Reader nicht ganz korrekt ist, werde ich im Verlauf dieses Buches diesen Begriff nicht mehr weiterverwenden. Stattdessen verwende ich ab jetzt den in der einschlägigen Literatur oft verwendeten Begriff Terminal.

Chipkartenterminals sind laut ZKA (Zentraler Kreditausschuss) in vier Klassen eingeteilt, wobei die nächste höhere Klasse jeweils die Funktionen der niederwertigeren Klassen mit einschließt:

Klasse	Beschreibung
1	Reine Kontaktiereinheit, mit Schnittstellenanbindung (RS232 oder USB) zu einer übergeordneten Einheit wie z.B. PCs
2	Wie Klasse 1 + Display
3	Wie Klasse 2 + Tastatur
4	Wie Klasse 3 + Sicherheitsmodul

Tabelle 3

Ein wichtiger Punkt ist die Anbindung eines Chipkartenterminals an das übergeordnete System, wobei es sich in unsrem Fall um einen PC handelt. Üblicherweise werden Chipkartenterminals per RS232-Schnittstelle oder per USB-Schnittstelle an einen PC angeschlossen. Während RS232-Terminals sofort nach dem Anschluss an den PC funktionsfähig sind, benötigen USB-Geräte in der Regel noch einen Gerätetreiber des jeweiligen Terminalherstellers, durch den sich dann auch die erweiterte Funktionalität eines höher Klassifizierten Chipterminals nutzen lässt.

Um ein Chipkartenterminal auf Softwareebene anzusprechen, gibt es verschiedene Möglichkeiten.
Die billigsten Terminals verfügen über nur wenig Eigenintelligenz und werden direkt über die RS232-Schnittstelle angesprochen. Dazu muss per Software einfach der COM-Port, an den das Terminal angeschlossen wurde, mit den entsprechenden Chipkartenparametern wie Baudrate, Start- und Stopbits, etc. geöffnet werden. Anschließend können die Daten von und zur Chipkarte übertragen werden. Der Vorteil dieses Verfahrens liegt darin, dass zur Kommunikation mit der Chipkarte keine weitere Software benötigt wird. Prinzipiell kann somit jedes beliebige Terminalprogramm verwendet werden, das eine serielle Kommunikation zweier Geräte über die RS232-Schnittstelle ermöglicht.
Der Nachteil liegt allerdings darin, dass diese Art der Kommunikation detaillierte Kenntnisse des RS232 Protokolls voraussetzt. Außerdem muss der COM-Port des Terminals, sowie die Übertragungsparameter (Baudrate, usw.) der Chipkarte bekannt sein, da sich diese sonst nicht ansprechen lässt. Ein weiterer Nachteil liegt darin, dass viele neuwertigere PCs nicht mehr über eine serielle Schnittstelle verfügen, so dass ein spezielles Konverterkabel verwendet werden muss, welches nicht immer alle Funktionalitäten einer seriellen

Schnittstelle bereitstellt. Im Allgemeinen wird diese Art der Kommunikation deshalb nur noch in speziellen Einzelfällen oder zu Diagnose verwendet.

Höherwertige Terminals werden über standardisierte Softwareschnittstellen angesprochen, durch diese sich die Kommunikation mit den Chipkartenterminals erheblich vereinfachen lässt. Die bekanntesten Softwareschnittstellen sind PC/SC, OCF, CT-API, MKT und MUSCLE. Durch die Verwendung einer Softwareschnittstelle kann mit jedem beliebigen Chipkartenterminal, unabhängig von seiner Hardware, über standardisierte Kommandos kommuniziert werden. Vorraussetzung dafür ist allerdings ein Gerätetreiber des Terminalherstellers, der die entsprechende Softwareschnittstelle implementiert. Kurz gesagt ist durch die Verwendung einer Softwareschnittstelle ein standardisierter, terminalunabhängiger Zugriff auf Chipkarten möglich.

1.4 Chipkartenprogrammierung

In vielen einschlägigen Internetforen liest man des Öfteren Fragenstellungen wie „Wie Programmiere ich Karte XY“ oder „Welches Programmiergerät brauche ich für Karte YXZ„.Wenn dem Fragesteller dann geantwortet wird, dass er dazu eine Entwicklungsumgebung des Kartenherstellers benötigt, mit der man Chipkartenprogramme in der Programmiersprache C, Java oder Basic erstellen kann, versteht der Fragesteller gar nichts mehr, denn eigentlich wollte der Fragesteller nur wissen, wie man eine Datei X auf seine Karte kopieren kann.
Irrtümlicherweise wird das Kopieren einer Datei auf eine Chipkarte, landläufig als „Programmieren“ bezeichnet, zu dessen Durchführung man ein spezielles Programmiergerät benötigt. Die Bezeichnung Programmieren wird in diesem Zusammenhang wohl deshalb so oft verwendet, weil im Bereich der Elektronik das Beschreiben eines EEPROMs als EEPROM-Programmierung bezeichnet wird, zu dessen Durchführung man tatsächlich ein spezielles Programmiergerät benötigt. Die Bezeichnung Programmieren ist aber in diesem Zusammenhang irreführend, denn Programmieren bedeutet nicht, eine fertige Binärdatei auf eine Chipkarte oder ein EEPROM zu kopieren. Vielmehr bezeichnet der Begriff Programmierung im engeren Sinn, die Tätigkeit, ein Computerprogramm mittels einer Programmiersprache zu erstellen. Im weiteren Sinn, sind mit der Bezeichnung Programmierung alle Tätigkeiten inbegriffen, die zur Erstellung des fertigen Computerprogramms führen. Üblicherweise bezeichnet man dies dann nicht mehr als Programmierung, sonder als Softwareentwicklung, denn die Programmcodierung ist nur ein einzelner Teilbereich innerhalb des kompletten Entwurfszyklus.
Vielleicht sollte ich hier noch kurz erwähnen, dass Sie zur Erstellung von Chipkartensoftware natürlich kein spezielles „Programmiergerät“ benötigen. Das einzige was Sie benötigen ist ein Terminal.

Wie Ihnen sicher bekannt sein dürfte, gibt es im PC Bereich zwei unterschiedliche Programmtypen: Einmal das Betriebssystem, welches Trivial gesagt die Hardware (I/O-Geräte, Speicher, usw.) eines Computers verwaltet und die Ausführung von Programmen steuert. Der zweite Programmtyp sind die Anwendungprogramme die ein Arbeiten mit dem PC überhaupt erst ermöglichen, denn ohne Anwendungssoftware könnten Sie keine Texte verfassen, Bilder zeichnen, Musik hören oder ins Internet gehen. Genau so verhält es sich mit einer Chipkarte. Ohne Anwendungssoftware würde der Mikrocontroller zwar funktionieren –genau wie ein PC ohne Anwendungssoftware funktionieren würde-, aber eine Chipkarte soll ja eine, von Ihnen bestimmte Aufgabe erfüllen.

In diesem Buch möchte ich Ihnen zeigen, wie Sie eigene Anwendungsprogramme für eine Chipkarte –in unserem Fall die BasicCard- erstellen können. Um ein Anwendungsprogramm für eine Chipkarte zu erstellen, benötigen Sie in der Regel weitere Hilfsmittel, die zumeist aus verschiedenen Softwaretools bestehen und in der Regel in einer so genannten IDE zusammengefasst (integriert) sind. IDE ist der englische Begriff für *integrated development environment,* womit ein Computerprogramm bezeichnet wird, das alle Komponenten enthält, die Sie zur Entwicklung eines Chipkartenprogramms benötigen. Oftmals wir anstatt dem Begriff IDE auch der Begriff SDK (Software Developement Kit) verwendet, womit aber dasselbe gemeint ist. Die IDEs bzw. SDKs können Sie entweder direkt vom Chipkartenhersteller oder von Drittanbietern beziehen. Allerdings möchte ich hier Anmerken, dass viele Hersteller ihre Entwicklungsumgebungen nicht kostenlos zur Verfügung stellen, sondern nur käuflich zu erwerben sind. Glücklicherweise hat der Hersteller der BasicCard einen anderen Weg eingeschlagen und stellt eine kostenlose IDE zur Verfügung, mit der wir uns in den nächsten Kapiteln herumschlagen dürfen.

1.5 Terminalprogrammierung

In der Kette Terminal, Chipkarte, Chipkartensoftware fehlt uns jetzt noch ein einzelnes Glied, nämlich die Terminalsoftware. Neben einem Chipkartenprogramm (Anwendungsprogramm auf der Chipkarte) benötigt man immer auch eine Terminalsoftware, welche mit dem Anwendungsprogramm auf der Chipkarte kommuniziert. Das Terminalprogramm sendet Anfragen an die Chipkarte, wertet die Antworten derselben aus und stellt diese in einer für den Benutzer mehr oder weniger ansprechenden Programmoberfläche dar. Die Terminalsoftware läuft in der Regel auf einem PC, auch wenn dieser manchmal zunächst für den Benutzer nicht sichtbar ist, wie z.B. bei einem Bankautomaten. In Abbildung 3 sehen Sie nun noch mal alle beteiligten Komponenten die bei der Chipkartenprogrammierung von Bedeutung sind.

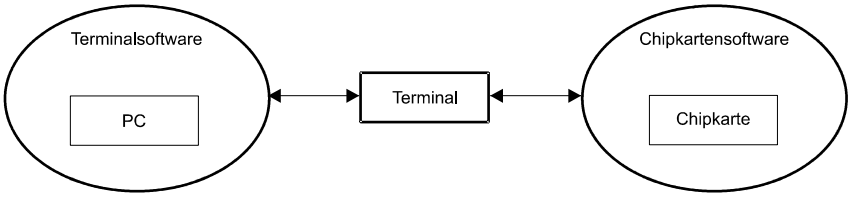

Abbildung 3

Für Sie als Chipkartenprogrammierer bedeutet dies, dass Sie neben der Chipkartensoftware also auch die Terminalsoftware erstellen müssen, wobei Ihnen zwei Möglichkeiten zur Auswahl stehen. Mit der BasicCard IDE können Sie einfache Text-Konsolenanwendungen entwickeln, die im so genannten DOS-Fenster ablaufen. Hierbei möchte ich aber gleich vorwegnehmen, dass mir diese Möglichkeit in Zeiten der GUIs nicht mehr Zeitgemäß erscheint, denn falls Sie beabsichtigen Ihr Programm später einmal zu verkaufen, ist der zukünftige Benutzer von Ihrer Programmoberfläche sicher nicht besonders Angetan, da man heutzutage eine komfortabel zu bedienende Windows-Anwendung erwartet hat.
Das erstellen von Terminalprogrammen unter Windows ist allerdings keine triviale Angelegenheit und setzt voraus, dass Sie sich mit der .NET Framework Programmierung auskennen und mindestens eine der NET Programmiersprachen wie z.B. C# beherrschen. Da dieses Buch eine Einführung in die Programmierung der BasicCard sein soll, werde ich das Thema Terminalprogrammierung unter Windows nur in einer kleinen Einführung behandeln, in der ich Ihnen die Grundlegenden Schritte aufzeige.

Bevor Sie nun dieses Buch wutentbrannt in die Ecke werfen, möchte ich Ihnen sagen, dass ein Terminalprogramm in Konsolenausführung auch seine Berechtigung hat, denn man kann mit wenigen Programmzeilen testen, ob das Chipkartenprogramm auch wirklich funktioniert. Insofern ist ein Terminalprogramm in Konsolenausführung für Diagnosezwecke sogar unentbehrlich.
Ein weiterer Grund besteht darin, dass man zum Programmieren zunächst gar kein Terminal und keine BasicCard benötigt, denn in der BasicCard IDE ist ein so genannter „virtueller Cardreader“ implementiert, der das vorhanden sein eines Terminals und einer darin eingesteckter BasicCard simuliert. Das Terminalprogramm in Konsolenausführung kann auf diesen „virtuellen Cardreader“ und auf die virtuelle BasicCard nun so zugreifen, als ob diese physikalisch vorhanden wären.
Für Sie als Programmierer ergibt sich dadurch den Vorteil, dass Sie sich zunächst gar keine zusätzliche Hardware (Terminal und BasicCard) beschaffen müssen, vielmehr können Sie Ihre Chipkarten- und Terminalprogramme einfach innerhalb der BasicCard IDE simulieren. Außerdem ersparen Sie sich in der Entwicklungsphase etwas Zeit, denn Sie müssen Ihre Chipkartenprogramme nicht jedes Mal auf eine reale BasicCard kopieren um diese zu testen.

Zum Schluss dieses Kapitels möchte ich Sie der Vollständigkeit halber noch darauf Hinweisen, dass eine OpenSource API für die BasicCard erhältlich ist, mit der Terminalprogramme für Linux erstellt werden können. Da ich mich mit diesem Thema leider noch nicht beschäftigen konnte, behalte ich mir vor, dieses Kapitel in einer weiteren Auflage dieses Buches zu behandeln.

1.6 Chipkartenkommandos

Die Kommunikation zwischen Terminalprogramm und Chipkartenprogramm erfolgt immer nach dem so genannten *Command and Response* Verfahren, wobei das Terminalprogramm der aktive und das Chipkartenprogramm der passive Kommunikationspartner ist. Beim *Command and Response* Verfahren sendet das Terminalprogramm Anfragen an die Chipkarte (an das Chipkartenprogramm), die daraufhin mit den entsprechenden Daten antwortet. Eine Chipkarte kann also niemals ein aktiver Kommunikationspartner sein, denn eine Chipkarte antwortet immer nur auf entsprechende Anfragen seitens des Terminals.

Alle Anfragen seitens des Terminals an die Chipkarte werden über so genannte Kommandos abgewickelt, wobei Ihre Aufgabe als Chipkartenprogrammierer darin besteht, diese Kommandos auf der Chipkarte zu implementieren. Möchte ein Terminal eine Anfrage an die Chipkarte stellen, so sendet das Terminalprogramm ein entsprechendes Kommando an die Chipkarte. Ist auf der Chipkarte das Kommando implementiert, antwortet diese mit den entsprechenden Daten. Man kann sich ein Kommando als eine Art Unterprogramm vorstellen, das auf der Chipkarte gespeichert ist, wobei das Terminalprogramm das gewünschte Unterprogramm auf der Chipkarte aufruft. Wenn Sie Beispielsweise eine Pinabfrage erstellen wollen, könnten Sie auf der Chipkarte ein Kommando (Unterprogramm) mit dem Namen *CheckPin* erstellen, das dann vom Terminalprogramm aufgerufen werden kann:

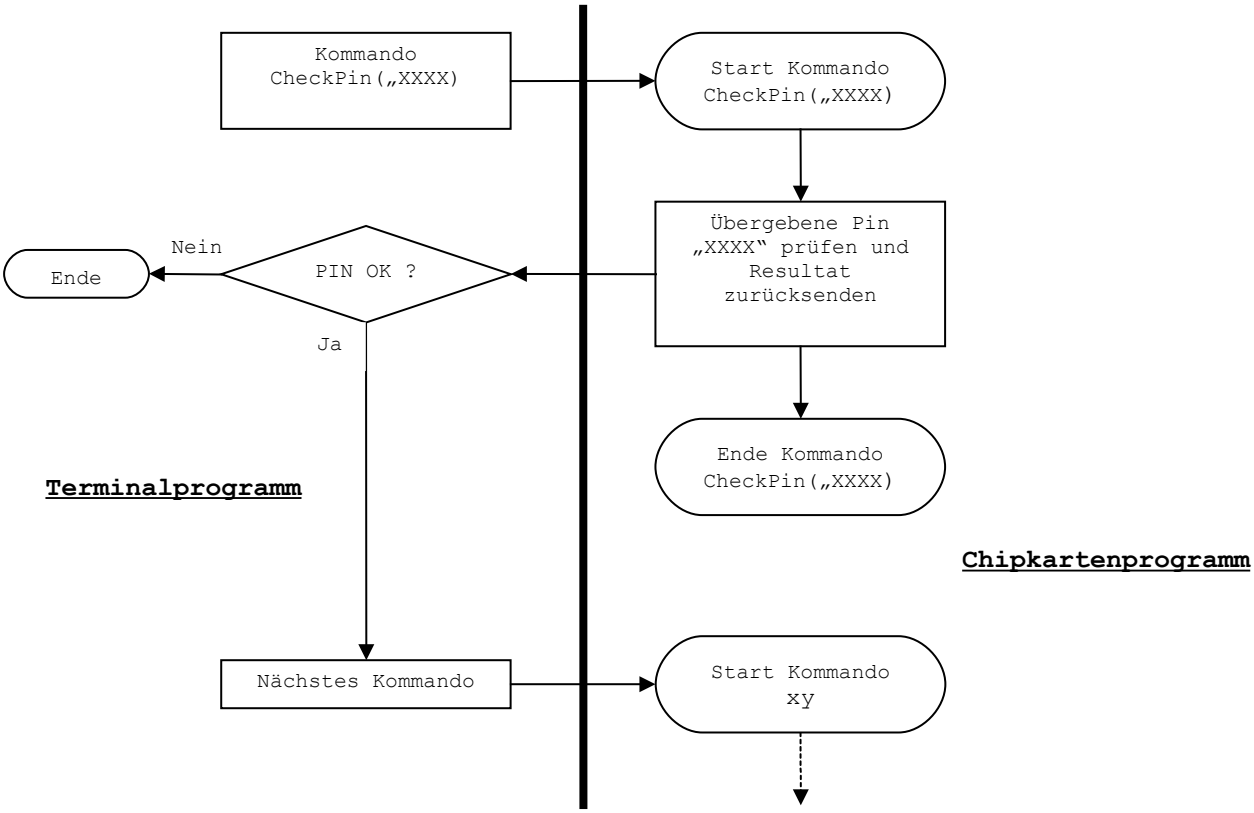

Abbildung 4

Das Terminalprogramm ruft das Kommando *CheckPin* auf der Chipkarte auf und übergibt diesem Kommando gleichzeitig die zu prüfende PIN, die in der Abbildung als „XXXX" dargestellt ist. Innerhalb des Kommandos *CheckPin* wird die übergebene PIN mit der auf der Karte gespeicherten PIN verglichen und das Prüfergebnis an das Terminalprogramm zurückgegeben. Das Terminalprogramm kann nun das Prüfergebnis auswerten und entsprechend reagieren, in dem z.B. die Programmausführung bei einer falschen PIN abgebrochen wird. War das Prüfergebnis positiv, d.h. die PIN war korrekt, wird das Terminalprogramm fortgesetzt und das nächste Chipkartenkommando aufgerufen.

1.6.1 Kommandostruktur

Wenn Sie schon einmal ein Computerprogramm auf dem PC geschrieben haben, wissen Sie sicher dass man Prozeduren, Funktionen, Methoden, usw. mit einem definierten Namen aufruft. In einem C# Programm würde man nachfolgende CheckPin-Methode (Methoden in C# bilden das Gegenstück zu Prozeduren und Funktionen in prozeduralen, d.h. nicht objektorientierten Programmiersprachen) mit
`bool bResult = CheckPin(strPin)` aufrufen.

```
// CheckPin Methode
bool CheckPin(string strString)
{
:
:
return bResult;
}
```

Auch innerhalb eines Terminalprogramms wird ein BasicCard-Kommando über den Kommandonamen aufgerufen, allerdings unterscheidet sich die Definition eines Chipkartenkommandos von herkömmlichen Funktionen in einem ganz bestimmten Punkt, denn zusätzlich zum Kommandonamen muss jedes Chipkartenkommando vom Programmierer mit einer zwei Byte Lange ID gekennzeichnet werden. Diese ID dient zu dem Zweck, dass nicht der komplette Kommandoname zur Chipkarte übertragen werden muss, sondern nur die zwei Byte Lange ID. Innerhalb des Terminalprogramms wird das Chipkartenkommando ganz gewöhnlich mit dem Kommandonamen aufgerufen, wenn Sie sich aber den Datenverkehr zwischen Terminalprogramm und Chipkarte anzeigen lassen, sehen Sie dass bei dem Kommandoaufruf nur die von Ihnen festgelegte Kommando-ID übertragen wird, und nicht der komplette Kommandoname.

Jedes Chipkartenkommando wird mit einer 2 Byte langen ID, sowie einem Kommandonamen definiert, wobei die ID und der Kommandoname vom Programmierer festgelegt werden müssen.

Neben der Kommando-ID werden noch weitere Daten zur Chipkarte gesendet, die für uns aber zum jetzigen Zeitpunkt noch nicht von Bedeutung sind. Stattdessen wollen wir die Kommando-ID etwas genauer unter die Lupe nehmen. Eine Kommando-ID besteht aus zwei Bytes, wobei das erste Byte als Class-Byte (CLA) und das zweite als Instruction-Byte (INS) bezeichnet wird. Das CLA-Byte wird dazu verwendet, Kommandos anhand ihres Verwendungszwecks zu gruppieren (klassifizieren) um damit die Übersichtlichkeit über die vom Programmierer verwendeten Kommandos zu erhöhen. Mit dem INS-Byte wird das eigentliche Kommando codiert.
Zur Veranschaulichen möchte ich Ihnen folgenden Vergleich zeigen: Eine Firma hat eine Datenbank in der die am Lager vorhandenen Artikel gespeichert. Die Datenbank, bzw. ein Datensatz hat folgenden Aufbau:

Artikel ID		Artikelbezeichnung	Lagerbestand
Artikelgruppe	Artikelkennung		

Tabelle 4

Jeder Artikeleintrag in der Datenbank besteht aus der Artikel ID und der Artikelbezeichnung, wobei die Artikel ID wiederum aus zwei Teilen besteht; der Artikelgruppe und der Artikelkennung. Die Datenbank könnte nun folgende Artikel beinhalten:

Artikel ID		Artikelbezeichnung	Lagerbestand
Artikelgruppe	Artikelkennung		
001	001	Motor 1 komplett	12
	002	Kolben	33
	003	Vergaser	342
	004	Nockenwelle	21
		usw.	54
002	001	Motor 2 komplett	543
	002	Kolben	34
	003	Vergaser	426
	004	Nockenwelle	76
		usw.	45
003	001	Motor 3 komplett	46
	002	usw.	454

Tabelle 5

Wenn nun ein Benutzer der Datenbank den Lagerbestand eines bestimmten Artikels abrufen möchte, könnte dieser in einer Eingabemaske die Artikel ID des gewünschten Artikels eingeben. Die Eingabe von 002003 würde beispielsweise den Artikelbestand des Artikels „Vergaser anzeigen“ (426 Stück). In diesem Beispiel wird deutlich, dass die Artikelbezeichnung nur für Informationszwecke von Bedeutung ist, denn zum Zugriff auf einen Artikel wird nur dessen Artikel ID benötigt.

Schauen wir uns nun den Aufbau eines Chipkartenkommandos an:

Kommando ID		Kommandoname
CLA-Byte	INS-Byte	

Tabelle 6

Abgesehen von der Spalte Lagerbestand, weist der Aufbau eines Chipkartenkommandos dieselbe Struktur wie die Artikeldatenbank auf. Die Kommando ID entspricht der Artikel ID, das CLA-Byte der Artikelgruppe und das INS-Byte der Artikelkennung. Als Chipkartenprogrammierer erstellen bzw. definieren Sie ein Chipkartenkommando, indem Sie diesem eine eindeutige Kommando-ID und einen Kommandonamen vergeben. Den Kommandonamen benötigen Sie nur während der Programmentwicklung, denn über diesen Namen wird im Terminalprogramm das Chipkartenkommando aufgerufen. Zur Laufzeit des Programms, d.h. wenn das Programm ausgeführt wird, wird aber nur die Kommando ID zu Chipkarte übertragen. Der Kommandoname dient also wie die Artikelbezeichnung im obigen Beispiel nur zu Informationszwecken - in diesem Fall für den Programmierer.

1.6.2 Rückgabewerte von Kommandos

Wie auch in anderen Programmiersprachen können an ein Chipkartenkommando Daten übergeben und von diesem an den Aufrufer - in diesem Fall das Terminalprogramm - wieder zurückgeben werden. Ob ein Chipkartenkommando Daten entgegen nehmen kann bzw. zurückgibt, wird vom Programmierer des Chipkartenkommandos festgelegt. Im Gegensatz zu anderen Programmiersprachen muss ein Chipkartenkommando neben den optionalen Daten immer einen so genannten Returncode zurückgeben, der dem aufrufenden Terminalprogramm mitteilt ob das Kommando erfolgreich ausgeführt wurde. Dieser Returncode wird als Statusword bezeichnet und besteht aus zwei Bytes, dem Statusword 1 (SW1) und dem Statusword 2 (SW2).

Jedes Chipkartenkommando muss eine Antwort in Form des Statuswords SW1SW2 an das Terminalprogramm zurücksenden.

Die Bedeutung vieler Returncodes ist in der ISO/IEC 7816-4 Norm festgelegt. Viele Chipkartenanwendungen verwenden aber eine von der Norm abweichende Codierung der Returncodes, was an sich kein Problem darstellt, solange dem Terminalprogramm bekannt ist, welche Returncodes verwendet werden und wie es diese auswerten muss. Die Returncodes die ein Chipkartenprogramm zurückgibt, können vom Chipkartenprogrammierer beliebig festgelegt werden. Der einzige Returncode, der quasi bei allen Chipkartenanwendungen dieselbe Bedeutung hat, ist der Returncode 9000 (SW1=90, SW2=00), der eine erfolgreiche Ausführung eines Kommandos signalisiert.

Betrachten Sie nun bitte nochmals den beispielhaften Ablauf des *CheckPin* Kommandos in Abbildung 4. Innerhalb des Kommandos *CheckPin* wird geprüft, ob die vom Terminalprogramm übergebene PIN korrekt ist und anschließend wird das Prüfresultat an das Terminalprogramm zurückgegeben.
In Abbildung 5 sehen Sie den korrigierten Ablaufplan des *CheckPin* Kommandos, das nun das Statusword SW1SW2 als Resultat (Antwort) an das aufrufende Terminal zurückgibt. Als Resultat wird in SW1SW2 = 9000 zurückgegeben wenn die PIN korrekt war (Sie erinnern sich: 9000 als Returncode bedeutet Kommando erfolgreich ausgeführt) und beispielsweise SW1SW2 = 9001 wenn die PIN nicht korrekt war. Das Terminalprogramm kann nun die Antwort (Statusword SW1SW2) des *CheckPin* Kommandos auswerten und darauf entsprechend reagieren.

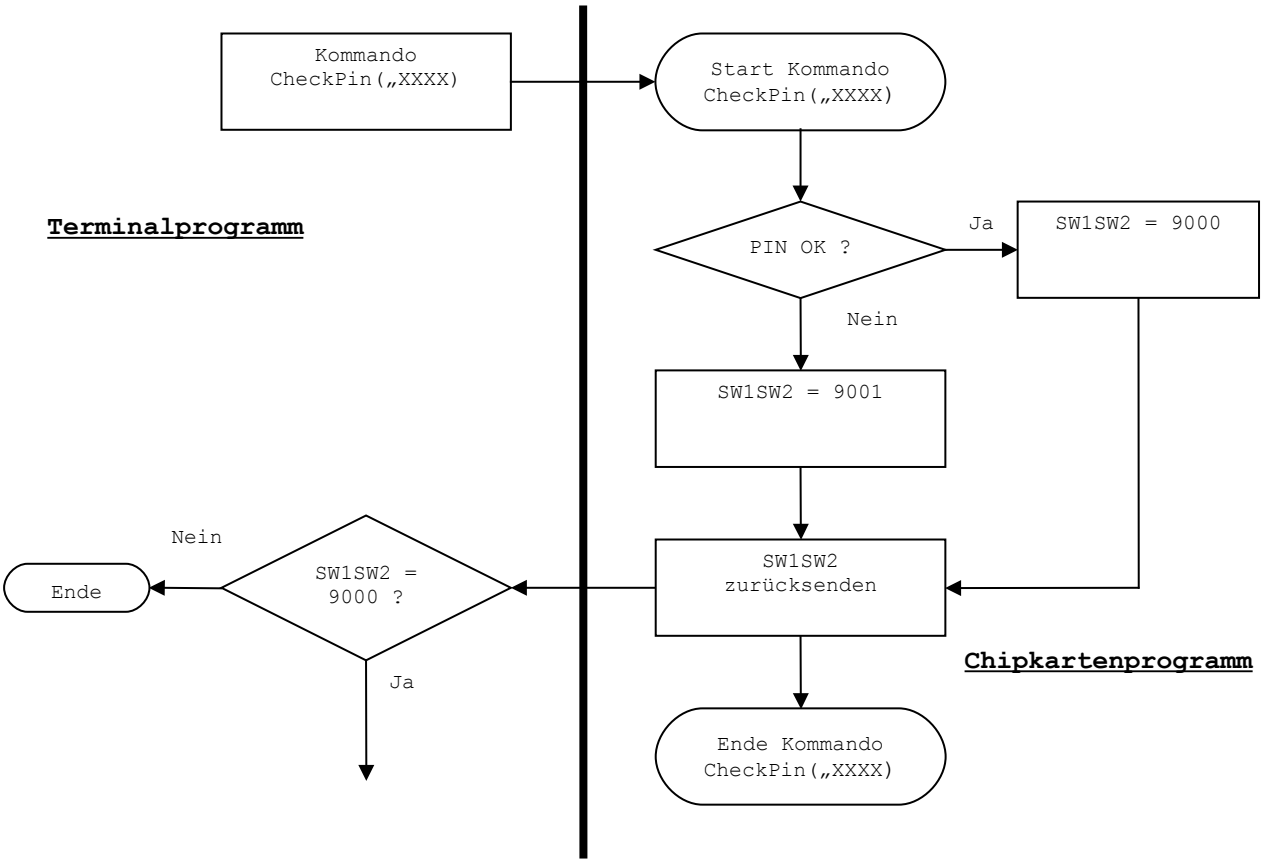
Kommando
CheckPin(„XXXX)
Start Kommando
CheckPin(„XXXX)
Terminalprogramm
PIN OK ?
Ja
SW1SW2 = 9000
Nein
SW1SW2 = 9001
Nein
Ende
SW1SW2 =
9000 ?
SW1SW2
zurücksenden
Ja
Chipkartenprogramm
Ende Kommando
CheckPin(„XXXX)

Abbildung 5

1.7 Genormte Chipkarten und Betriebssysteme

In vielen Bereichen unseres täglichen Lebens benutzen wir technische Produkte, deren Herstellung und Funktionsweise auf einheitliche Standards basieren. Durch diese Vorschriften wie etwas Hergestellt oder Funktionieren soll, wird sichergestellt dass diese Produkte untereinander austauschbar sind. Um dies zu erreichen, werden die grundlegenden Eigenschaften und Merkmale eines Produkts von zentralen Stellen (den Normgremien) in einem Normregelwerk festgelegt.

Für die Herstellung, den Aufbau und die Funktionsweise einer Chipkarte gibt es –wie auch in anderen Bereichen- eine beachtliche Anzahl von Normgremien, eine Vielzahl der daraus resultierenden Normen, und weitere Ableitungen dieser Normen. Für Chipkarten mit Mikrocontroller, ist überwiegend die ISO-Norm 7816 von Bedeutung, die eine Erweiterung der ISO-Norm 7810 darstellt. Die ISO-Norm 7816 ist eine mehrteilige Norm zur Spezifikation der wesentlichen Merkmale von kontaktlosen und Kontaktbehafteten Chipkarten, wobei kontaktlose Chipkarten wiederum in anderen Normen detaillierter behandelt werden. In der nachfolgenden Tabelle finden Sie eine Auflistung der wichtigsten Teile der ISO-Norm 7816 für kontaktbehaftete Chipkarten:

Teil	Inhalt	Anmerkung
1	Physikalische Eigenschaften	Hier sind die maximalen Grenzwerte wie z.B. Temperatur, UV-Strahlung, Biegefaktor, Kontaktverschleiß, usw. festgelegt, bevor die Chipkarte ihren Dienst versagt.
2	Abmessungen, Position und Größe der Kontaktfläche und des Mikrocontrollers	Hier sind die zulässigen Abmaße und Positionen des Mikrocontrollers und dessen Kontaktfläche festgelegt.
3	Elektrische Eigenschaften und Übertragungsprotokolle	Hier sind alle elektrischen Eigenschaften wie Spannungsversorgung, Taktfrequenz, physikalische Übertragungsprotokolle, usw. festgelegt.
4	Kommandos für Anwendungen	Hier sind einheitliche Chipkartenkommandos, Kommandostrukturen und Returncodes, sowie die Dateiorganisation und Zugriffsmethoden auf Chipkartendateien festgelegt.
6	Datenelemente	Hier sind die Datenobjekte, auf die mit den Kommandos aus Teil 4 zugegriffen werden können, festgelegt.
7	Strukturierte Kartenabfragesprache SCQL	SCQL ist eine an SQL angelehnte Datenbanksprache mit der auf eine Daten-bank, die auf der Chipkarte gespeicherten ist, zugegriffen werden kann. Der Aufbau der Datenbank, die Kommandos und der SCQL-Syntax ist hier festgelegt.
8	Kommandos für Sicherheit	Hier sind alle Chipkartenkommandos und Returncodes, festgelegt, die für sicherheits-relevante Bereiche (Verschlüsselung, sichere Kommunikation, Hashwertberechnungen, usw.) von Bedeutung sind.
9	Erweiterte Kommandos	Hier sind weitere Chipkartenkommandos und Returncodes festgelegt, die nicht in Teil 4 enthalten sind.
13	Kommandos für „Multi-Application“ Chipkarten	Auf einer „Multi-Application“ Chipkarte können gleichzeitig mehrere Anwendungs-programme vorhanden sein. In diesem Abschnitt sind Chipkartenkommandos und Returncodes festgelegt, mit denen sich diese Anwendungen verwalten lassen.

Tabelle 7

Bitte beachten Sie, dass der Begriff ISO 7816 nicht für eine einzelne Norm steht, sondern als übergeordneter Begriff für die einzelnen Teilnormen steht. Bei Chipkarten wird deshalb immer mit angegeben, welche Teile der Norm die Chipkarte erfüllt. Die meisten Chipkarten werden nach den Spezifikationen der ISO-Normen 7816 - Teil 1 bis Teil 3 produziert, in denen die Grundlegenden physikalischen Funktionen der Chipkarte festgelegt sind. Je nach Verwendungszweck können diese Chipkarten anschließend in Ihrer Funktionalität gemäß der ISO-Norm 7816 – Teil 4 bis Teil 13 erweitet werden, wobei üblicherweise nicht alle Teile der Norm in die Chipkarte integriert werden, sondern nur die für den Verwendungszweck benötigten.

Während sich die Normteile 1-3 größtenteils auf die Hardware von Chipkarten beziehen, beschreiben die Normteile 4-13 die Anwendungssoftware von Chipkarten. Chipkarten nach ISO 7816-3 und höher können im Gegensatz zu frei programmierbaren Chipkarten wie beispielsweise die BasicCard in der Regel nicht mehr mit extern erstellter Anwendungssoftware beschrieben werden, da bei diesen Chipkarten bereits eine oder mehrere Anwendungen bereits in das Chipkartenbetriebssystems (Card Operating System – COS) integriert wurden. Chipkarten mit integrierten Anwendungen stellen u. a. genormte Kommandos und Datenstrukturen zur Verfügung, anhand derer beispielsweise Dateien und Verzeichnisse auf der Chipkarte angelegt werden, Daten aus Dateien gelesen, oder in Dateien geschrieben werden. Des Weiteren sind auf diesen „fertigen“ Chipkarten standardisierte kryptografische Algorithmen implementiert, mit denen z.B. elektronische Signaturen oder Zertifikate erstellt oder überprüft werden können.
Prinzipiell müssen bei der Verwendung fertiger Chipkarten also nur noch die Terminalprogramme erstellt werden, die je nach Verwendungszweck die gewünschten Chipkartenkommandos aufrufen und die zurückgegebenen Daten auswerten. Das Resultat sind hochprofessionelle, sichere und standardisierte Anwendungen.

Nun stellt sich mancher Leser natürlich die Frage, weshalb man sich überhaupt die Mühe machen sollte, eigene Chipkartenanwendungen zu Programmieren, wenn doch bereits fertige Chipkarten mit ausgereiften und professionellen Chipkartenbetriebssystemen käuflich zu erwerben sind. Diese Frage ist natürlich berechtigt und in den meisten Fällen ist es auch sinnvoll eine fertige Chipkarte zu verwenden, denn Sie würden zum Schreiben eines Briefes ja auch nicht erst ein Textverarbeitungsprogramm programmieren. Wenn Sie aber eine bestimmte Anwendung benötigen, die noch nicht programmiert wurde, müssen Sie selbst in die Tasten greifen. Ebenso wie im Bereich der PC-Software gibt es eben auch im Bereich der Chipkartenprogrammierung immer wieder Aufgaben, die sich mit Standardsoftware nicht lösen lassen.
In manchen Fällen möchte man sogar ganz bewusst auf die Verwendung fertiger Chipkarten mit genormten Kommandos verzichten und seine eigene Proprietäre Chipkartenanwendung erstellen. Dies kann der Fall sein, wenn die Chipkarte für einen ganz speziellen Zweck eingesetzt werden soll. Als Beispiel seien hier die einzelnen Chipkartenprogramme von Pay-TV Chipkarten genannt

Ein weiterer Grund ist natürlich auch der Spaß am Programmieren und das Wissen, dem elitären Kreis der Chipkartenprogrammierer angehörig zu sein. ☺

Zum Abschluss dieses Kapitels möchte ich Ihnen noch einige Chipkartenbetriebssysteme, deren Hersteller und deren Internet-Adresse auflisten:

COS	Hersteller	WWW
CARDOS	Siemens	www.it-solutions.siemens.com/b2b/it/de/global/loesungen-services/business-solutions/identity-management-biometrics/smart-card-products/Pages/smart-card-products.aspx
STAROS	Giesecke & Devrient	www.gi-de.com/de/index.jsp
MULTOS	MAOSCO Limited	www.multos.com
SECCOS	EURO Kartensysteme	www.seccos.info
TCOS	TeleSec (Telekom)	www.telesec.de/tcos/index.html

Tabelle 8

Außerdem lohnt sich zu dieser Thematik ein Blick auf folgende Internetseiten:

www.cardwerk.com/smartcards/smartcard_standards.aspx
www.cryptoshop.com

1.8 Übertragungsprotokolle

Unter einem Übertragungsprotokoll, das auch als Kommunikationsprotokoll bezeichnet wird, versteht man ein Regelwerk in dem beschrieben wird, wie zwei Kommunikationspartner miteinander kommunizieren. Beide Kommunikationspartner müssen sich exakt an das Übertragungsprotokoll (Regelwerk) halten, da sonst keine Kommunikation möglich ist. Lassen Sie mich dies an einem Beispiel aus der realen Welt verdeutlichen:

Sie und Ihre Schwester sind in Polen geboren und aufgewachsen. Irgendwann ist Ihre Schwester dann nach Amerika und Sie nach Deutschland ausgewandert. Während Sie anfänglich noch des Öfteren in telefonischem Kontakt mit Ihrer Schwester standen, wurden die Anrufe mit der Zeit immer seltener, bis diese dann irgendwann ganz eingestellt wurden.
Nachdem nun 10 Jahre vergangen sind, seit Sie das letzte Mal Kontakt mit Ihrer Schwester hatten, beschließen Sie, dass Sie sich mal wieder melden könnten. Allerdings kennen Sie weder die Postanschrift, noch E-Mail-Adresse Ihrer Schwester. Das einzige woran Sie sich noch erinnern können ist die Telefonnummer, von der Sie hoffen, dass diese noch gültig ist.
Als weiters Problem könnte sich die Sprache erweisen. Sie sprechen kein Englisch und Ihre Schwester höchstwahrscheinlich auch kein Deutsch. Somit bleibt nur zu hoffen, dass Sie Ihrer beiderseitigen Muttersprache Polnisch noch mächtig sind.

Lassen Sie uns nun noch mal in Erinnerung rufen, was man unter einem Übertragungsprotokoll bzw. Kommunikationsprotokoll versteht:

Ein Übertragungsprotokoll ist ein Regelwerk in dem beschrieben wird, wie zwei Kommunikationspartner miteinander kommunizieren. Beide Kommunikationspartner müssen sich exakt an diese Regeln halten, da sonst keine Kommunikation möglich ist.

Wenn wir diese Aussage nun auf unser Beispiel abstrahieren, lässt sich folgendes erkennen: Damit wir erfolgreich kommunizieren können, benötigen wir ein Protokoll das sowohl die physikalische Verbindung (Telefon), sowie die sprachliche Verbindung (Polnisch) beinhaltet. Da die sprachliche Verbindung erst möglich ist, wenn die physikalische Verbindung zustande gekommen ist, können wir sagen, dass die sprachliche Verbindung auf der physikalischen Verbindung aufbaut. Unsere Protokollbeschreibung lautet also:

„In der unteren Protokollebene wird beschrieben wie die physikalische Verbindung (Telefon, Brief, Fax, etc.) erfolgen soll. In der oberen Protokollebene wird beschrieben wie die sprachliche Verbindung zustande kommen soll. Beide Kommunikationspartner müssen sich exakt an das Protokoll (Ebene 1 und 2) halten, da sonst keine Kommunikation möglich ist."

Der Ablaufplan dieser Protokollbeschreibung ist in Abbildung 6 dargestellt.

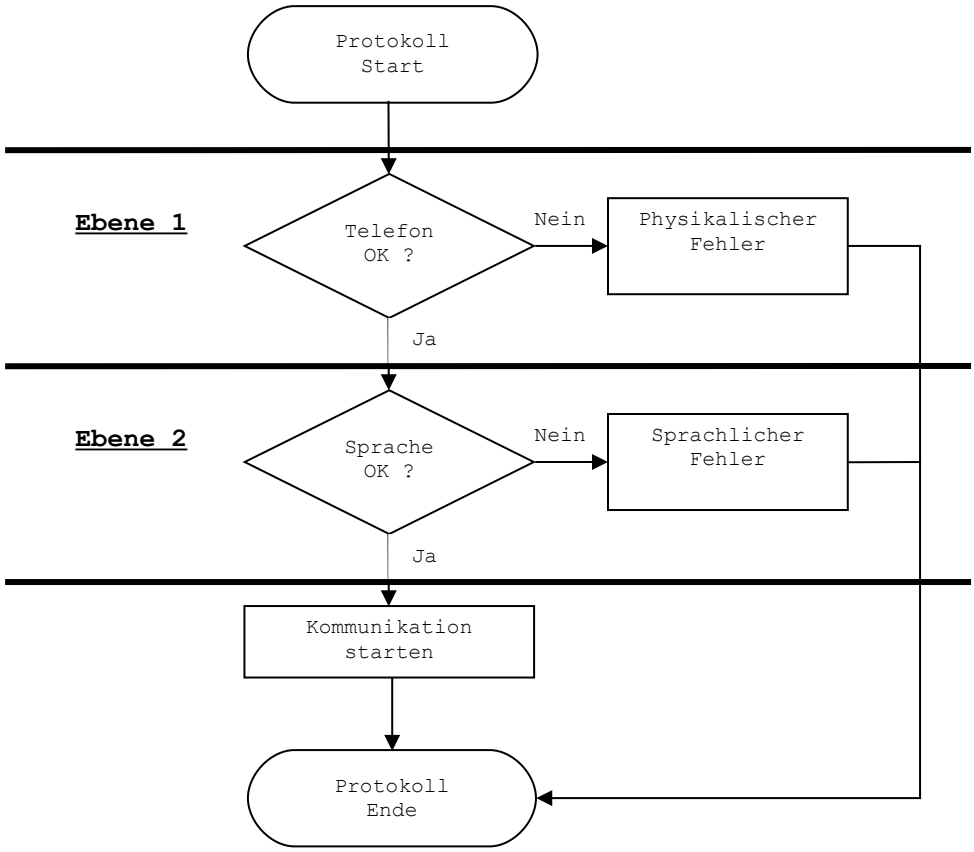

Abbildung 6

Nachdem wir nun wissen, was Übertragungsprotokolle sind und wozu diese verwendet werden, wollen wir uns die bei Chipkarten verwendeten Übertragungsprotokolle etwas näher anschauen.
Das Protokoll, das bei der Kommunikation zwischen Terminal und Chipkarte eingesetzt wird, besteht wie in unserem Beispiel, aus zwei aufeinander aufbauenden Protokollebenen. Während auf der unteren physikalischen Ebene die hardwarespezifischen Eigenschaften wie z.B. Spannungspegel, Taktfrequenzen usw. beschrieben werden, werden auf der oberen logischen Ebene die softwarespezifischen Übertragungsformate spezifiziert. Die Datenübertragung auf physikalischer Ebene erfolgt üblicherweise nach dem bereits in Kapitel 1.2.2 genannten RS232-Protokoll. Für die Datenübertragung auf logischer Ebene werden sogenannte Tx Übertragungsprotokolle verwendet, wobei kontaktbehaftete Mikrocontrollerkarten überwiegend die in ISO-Norm 7816-3 beschriebenen Übertragungsprotokolle T0 und T1 verwenden.

T0 Protokoll

Das T0 Protokoll gilt als das erste genormte Übertragungsprotokoll für kontaktbehaftete Chipkarten, das auch heute noch große Anwendung findet, aber zunehmend vom moderneren T2 Protokoll verdrängt wird. Das T0 Protokoll zeichnet sich durch seinen geringen Ressourcenverbrauch sowie seinem sehr einfachen Aufbau aus. Nachteilig zu nennen, wäre die etwas umständlich zu handhabende gesicherte Datenübertragung zwischen Terminal und Chipkarte, sowie die eingeschränkte Fehlerkorrektur, die sich ausschließlich auf die Auswertung des Paritätsbits beschränkt.
Aus technischer Sicht betrachtet ist das T0 Protokoll ein asynchrones Protokoll, das Daten im halbduplex Betrieb (Wechselseitige Nutzung der Übertragungsleitung) sendet und empfängt.
Die Kommunikation zwischen Terminal und Chipkarte erfolgt byteorientiert, d.h. die kleinste Dateneinheit die während der Kommunikation übertragen werden kann, besteht aus einem Byte.

T1 Protokoll

Das T1 Protokoll gilt als Nachfolger des T0 Protokolls und basiert auf dem OSI-Schichtenmodell, wobei das T1 Protokoll der Sicherungsschicht (Layer 2 - Data Link Layer) entspricht. Die Hauptvorteile des T1 Protokolls gegenüber dem T0 Protokoll sind eine bereits integrierte gesicherte Datenübertragung zwischen Terminal und Chipkarte, sowie eine hoch entwickelte Fehlerkorrektur.
Auch das T1 Protokoll ein asynchrones Protokoll, das Daten im halbduplex Betrieb sendet und empfängt.. Die Kommunikation zwischen Terminal und Chipkarte erfolgt im Gegensatz zum T0 Protokoll nicht byte- sondern blockorientiert, d.h. die kleinste Dateneinheit die während der Kommunikation übertragen werden kann, besteht jeweils aus einem kompletten Datenblock.

Alle BasicCard Typen verwenden serienmäßig immer das T1 Protokoll, sofern nicht explizit vom Programmierer angegeben wird, dass das T0 Protokoll verwendet werden soll. Allerdings kann das T0 Protokoll nur von den BasicCard Typen „Professional“ und „MultiApplication“ verwendet werden. BasicCard Typen in der Ausführung „Enhanced“ unterstützen nur das T1 Protokoll.

Prinzipiell sollten Sie für Ihre Programme immer das voreingestellte T1 Übertragungsprotokoll verwenden, es sei denn, Sie wollen ein Chipkartenprogramm erstellen, das mit einem bereits bestehendem T0 Terminalprogramm verwendet werden soll, oder wenn Sie ein T0 Chipkartenprogramm nachprogrammieren möchten.

1.9 Die BasicCard

1.9.1 Allgemeines

Die BasicCard ist eine frei programmierbare ISO 7816-1/3 kompatible Prozessorkarte, die von der Firma „Zeitcontrol Cardsystems GmbH“ in unterschiedlichen Versionen und Typen angeboten wird.
Frei Programmierbar bedeutet, dass die BasicCard über keinen festgelegten Befehlssatz nach ISO 7816-4 verfügt, sondern selbst programmiert werden kann. Als Programmiersprache wird ein BASIC-Dialekt Namens ZC-Basic eingesetzt, dessen Syntax weitgehend an die früheren Basic-Dialekte angelehnt ist und um spezielle Chipkartenspezifische Funktionen erweitert wurde.

Der Name BasicCard bezieht sich auf das im ROM der BasicCard gespeicherte Chipkartenbetriebssystem (Card Operating System – COS), das einen Interpreter beinhaltet, der den vom Basic-Compiler erzeugten P-Code interpretiert. Die BasicCard unterstützt je nach Kartenversion und Kartentyp viele aktuelle kryptografische Verfahren wie z.B. DES oder AES, sowie weitere Verfahren, die bei Bedarf über Programmbibliotheken hinzugefügt werden können.

Während in früheren Versionen der BasicCard noch ATMEL Mikrocontroller aus der AVR Reihe eingesetzt wurden, werden in den aktuellen Versionen so genannte „Secure Smart Card Controller“ der Firma NXP aus der P5Cx Reihe sowie „Contact Smart Card ICs“ der Firma EM Microelectronic verwendet. Diese speziell für Chipkarten gefertigten Mikrocontroller zeichnen sich u. a. dadurch aus, dass in diese bereits eine Reihe von Mechanismen integriert wurden, die Angriffe auf physikalischer Ebene erschweren bzw. unmöglich machen sollen. Grundsätzlich werden bei allen auf den BasicCards eingesetzten Mikrocontrollern die Versorgungsspannung und die Taktfrequenz überwacht. Bei den höherwertigeren BasicCard Versionen kommen entsprechend höherwertigere Mikrocontroller zum Einsatz, die weitere Detektoren und Überwachungsmechanismen beinhalten. So enthält beispielsweise die BasicCard Professional ZC7.x Familie einen Mikrocontroller vom Typ NXP- P5CD040V0B, der sich für Anwendungen mit hohen Sicherheitsanforderungen geradezu empfiehlt. Weitere Informationen zu den verwendeten Mikrocontrollern finden Sie unter folgenden Internetseiten:

```
www.nxp.com/#/homepage/cb=[t=p,p=/53420/71108/71425]|pp=[t=pfp,i=71425]
www.nxp.com/#/homepage/cb=[t=p,p=/53420/71108/71425]%7Cpp=[t=pfp,i=71425]
www.emmarin.com/Line.asp?IdLine=7)
www.commoncriteriaportal.org/products/
```

Leider ist es für den privaten Softwareentwickler mit schmalem Geldbeutel, nicht ganz einfach eine einzelne BasicCard zu erwerben, denn die Firma Zeitcontrol bietet die BasicCard erst ab einer Mindestabnahmemenge von 10 Stück an, was für die meisten interessierten Einsteiger zuviel sein dürfte. Soweit mir bekannt ist, gibt es in Deutschland auch keinen weiteren Händler, bei dem man eine einzelne BasicCard erwerben könnte. Somit bleibt dem Leser nur die Möglichkeit gleich einen ganzen 10er Pack bei der Firma Zeitcontrol zu bestellen, oder sich nach einem Händler aus dem Ausland umzuschauen. Wenn Sie noch keinen Chipkarten-Terminal besitzen, haben Sie auch die Möglichkeit das relativ günstige „BasicCard Developement Kit“ der Firma Zeitcontrol für rund 70 Euro zu erwerben, in dem zwei Chipkartenterminals (Smartcard-Reader), drei BasicCards, sowie das SDK und die Dokumentation enthalten ist.

1.9.2 BasicCard Typen

Die BasicCard ist in drei verschiedenen Ausführungen erhältlich, und zwar in den Ausführungen Enhanced, Professional und MultiApplication. Von jeder Ausführung gibt es wiederum unterschiedliche Versionen, die sich hauptsächlich im Befehlsumfang und in der Hardwareausstattung unterscheiden. Nachfolgende Tabelle gibt eine kleine Übersicht über die aktuell erhältlichen BasicCards. Eine detailliertere Übersicht finden Sie im Anhang.

Ausführung	Version	EEPROM in KB	RAM in Bytes	Kryptografische Merkmale
Enhanced	ZC3.12	2	256	DES
	ZC3.13	2	256	DES
	ZC3.32	8	256	DES
	ZC3.33	8	256	DES
	ZC3.42	16	256	DES
	ZC3.43	16	256	DES
Professional	ZC5.4	16	1930	AES, DES,EAX,OMAC
	ZC5.5	32	1930	AES, DES,EAX,OMAC
	ZC5.6	60	1930	AES, DES,EAX,OMAC
	ZC7.5(1)	32	2912	AES, DES,EAX,OMAC
MultiApplication	ZC6.5	30	1520	AES, DES, EAX, OMAC

Tabelle 9

1) Auch als RFID und Dual Interface Ausführung (RFID + Kontakte) erhältlich

Mit der BasicCard IDE können Sie auch Programme für BasicCards erstellen, die von der Firma Zeitcontrol aktuell nicht mehr angeboten werden (Tabelle 10). Sollten Sie die Möglichkeit haben eine dieser Karten günstig erwerben zu können, können Sie bedenkenlos zugreifen, denn für viele Anwendungen genügen diese Karten vollauf.

Ausführung	Version	EEPROM in KB	RAM in Bytes	Kryptografische Merkmale
Enhanced	ZC3.2	4	256	DES
	ZC3.7	2	256	DES
	ZC3.8	4	256	DES
	ZC3.9	8	256	DES
Professional	ZC4.5A	30	1024	AES, RSA
	ZC4.5D	30	1024	DES, RSA

Tabelle 10