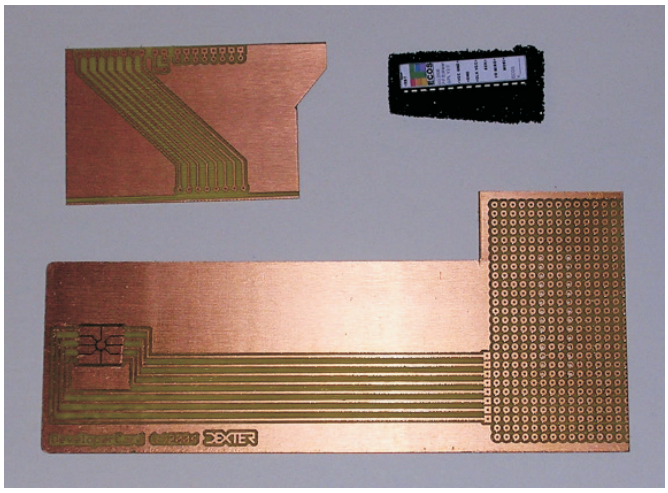
DeveloperCard Bausatz

Vielen Dank das Sie sich für einen DeveloperCard-Bausatz entschieden haben. Die DeveloperCard wurde speziell für Lehrzwecke entwickelt und gestattet Ihnen die Welt der Chipkarten Schritt für Schritt zu erforschen. Für die ersten Schritte benötigen Sie weder Programmierkenntnisse, noch müssen Sie über Protokolle und Ähnliches bescheid wissen. Ein normales Kartenlesegerät (Empfohlen: TowitokoChipdrive, seriell) und eine Software zum Senden von Befehlen (z.B.: Gscriptor) reichen für die ersten Schritte aus.

Das ist alles dabei:

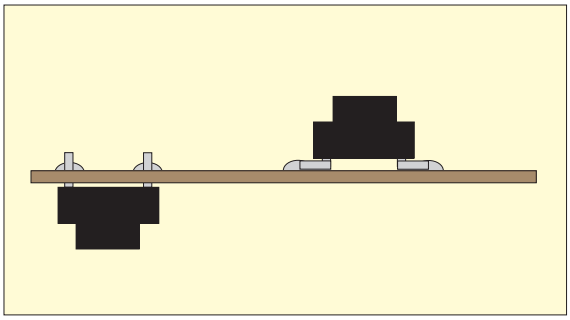


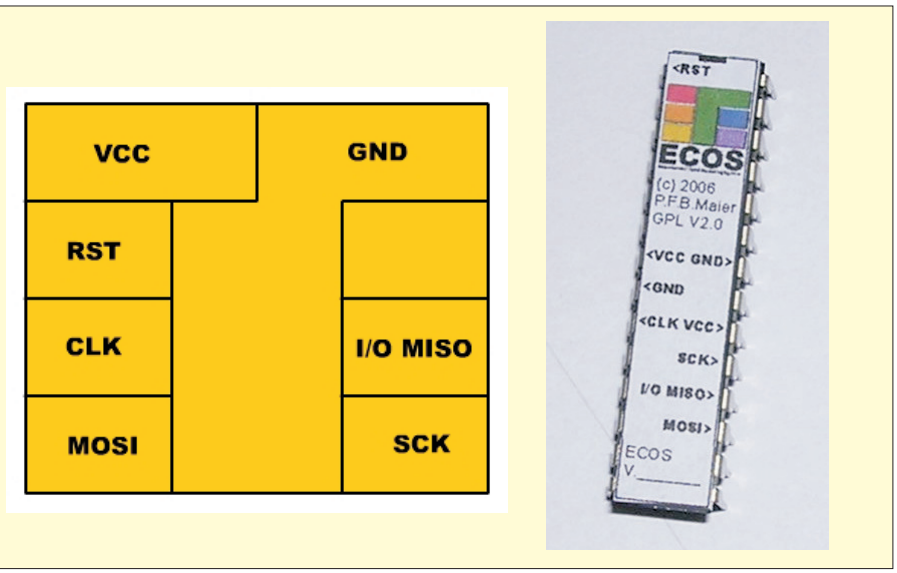
Zu dem Bausatz gehört zwei Platinen: Die große hat die Form einer Chipkarte und am hinteren Teil ein Lochraster mit dem Sie beliebige Schaltungen aufbauen können. Die zweite Platine wurde dafür ausgelegt eine Kontaktiereinrichtung für Chipkarten (Conrad-Bestellnummer: 730513-LN) aufzunehmen. Um z.B.: ein kleines Chipkartenlesegerät zu bauen. Die beiden Platinen können mit einander verbunden werden so das sie zu einer großen Platine werden. Mit dabei sind noch ein Microcontroller (ATmega8, Conrad-Bestellnummer: 154054-LN) auf dem das Kartenbetriebsystem XCOS bereits vorinstalliert. Nach dem zusammen bauen ist die Karte also sofort benutzbar. Das Kartenbetriebsystem ist in C geschrieben und kann individuell erweitert werden.

Der Controller wird mit der beiliegenden Federfassungen montiert so das er bequem ausgetauscht werden kann. Jetzt brauchen Sie nur noch etwas Fädeldraht und einen Lötkolben und es kann losgehen!

Und los gehts:

Nun wird es ernst: Sie müssen zuerst den Controller auf der Platine befestigen. Wie Sie sehen steckt der Controller schon in den Federfassungen, das ist auch richtig so. Sie haben zwei Möglichkeiten den Controller zu montieren: Sie können ihn durch die vorgebohrten Löcher von unten durch die Platine stecken und festlöten oder Sie biegen vorsichtig die Beine der Fassung im 90Grad Winkel zur Seite und löten den Controller auf der Oberseite fest. Ersteres wäre der professionellere Weg, letzteres ist aber praktischer da der Controller bei eingesteckter Karte oben liegt.





Nun müssen die Anschlüsse des Controllers mit den Anschlüssen der Chipkartenplatine verbunden werden. Die Grafik zeigt die Pinbelegungen des Controllers und die der Kontaktflächen auf der Karte. Alles muss 1:1 mit einander verbunden werden. Zu beachten ist das der Controller 2 VCC (+) und zwei GND (-) Anschlüsse hat. Diese müssen auch angeschlossen werden. Die Anschlüsse mit der Bezeichnung MOSI, MISO und SCK sind die Anschlüsse für das Programmiergerät. MISO liegt auf dem I/O Port der Chipkarte. Ansonsten läuft hier alles streng nach ISO7816.

Nun ist Ihre Karte betriebsbereit und sollte wenn sie in ein Kartenterminal gesteckt wird einen ATR (Answer to Reset aussenden und dann vom Lesegerät als SmartCard erkannt werden. Bei einem Towitoko-Chipdrive wird zudem das erfolgreiche Erkennen der Karte mit einer grünen LED quittiert. Der ATR der Karte kann mit dem Befehl pcsc_scan auf der Konsole angezeigt werden. Zum “Sprechen” mit der SmartCard wird das Program Gscriptor, welches in dem PCSC-Paket schon enthalten ist empfohlen.

Natürlich müssen Sie keinen Kartenleser verwenden um mit Ihrer Karte zu sprechen, Sie können, da Smartcards vom Prinzip her eine normale serielle Schnittstelle (nur beide Kanäle auf einer Leitung) besitzen auch die serielle Schnittstelle Ihres PCs verwenden (Pegelanpassung nicht vergessen!). Die Karte sendet ihre Bits mit einem Teiler von 372, das bedeutet das alle 372 Takte ein Bit gesendet wird. Wenn Sie einen Quarz mit 3,5712 Mhz verwenden ergibt sich eine Geschwindigkeit von 9600 Baud.

Der ATR:

Mit dem ATR begrüßt die SmartCard das Terminal, es ist die Art wie alle Smartcards “Hallo!” sagen. Der ATR gibt Auskunft darüber wie mit der Karte kommuniziert werden soll und enthält in der Regel noch ein par sogenannte “Historical Bytes”, das ist eine frei wählbare Zeichenkette.

Der ATR Ihrer DeveloperCard mit XCOS lautet: 3B 7F 13 00 FF 45 43 4F 53 76 30 31 30 28 63 29 44 58 54 2D

So unterhält man sich mit SmartCards:

Wenn Sie es geschafft haben - auf welchem Wege auch immer - den ATR der Karte zu empfangen können Sie versuchen mit ihrer Karte zu reden. Wie viele elektronische Dinge haben auch SmartCards ihre eigene Sprache (Protokolle). Verbreitet sind zwei: T=0, und T=1. T=1 ist Blockorientiert, es wird also immer eine Gruppe von Bytes gesendet und mit einer Checksumme überprüft. Da Blockprotokolle für Anfänger oft schwierig zu sprechen sind spricht unsere Karte T=0, ein simples byte-orientiertes Protokoll. Zu erwähnen ist auch das SmartCards sehr schüchterne Gesprächspartner sind, sie reden nur wenn sie gefragt werden. Selbst der ATR kommt nicht von alleine aus der Karte, sie müssen Spannung anlegen und die Reset-Leitung betätigen.

Bevor wir allerdings loslegen schauen wir uns zunächst einmal das T=0 Protokoll an:
Ein Befehl besteht im wesentlichen immer aus einer Gruppe von 5 Bytes. Das erste ist das sog. Class-Byte. Hiermit wird der Karte gesagt welche Gruppe von Befehlen gemeint ist. Bei XCOS sind Befehle die irgendwelche Informationen über die Karte ausgeben in der Gruppe “0A” Nach dem Class-Byte folgt sofort das Instruction-Byte, welches der eigentliche Befehel ist. Der Befehl “01” gibt bei XCOS die Betriebsystem-Version aus. Fassen wir also zusammen: “8A 01” Bedeutet: “Führe bitte den Versionsnummer-Infobefehl (01) aus der Gruppe mit den Info-Befehlen (0A) aus”. Nach dem wir erfolgreich den Befehl aufgerufen haben folgen zwei Parameterbytes (P1 und P2) Da der Befehl mit der Versionsnummer keine Parameter hat sind diese Bytes hier beide “00”. Als letztes Byte wird mit angegeben wie viele Bytes an Daten man von der Karte zurückbekommen möchte. Da die Versionsnummer 3 Byte lang steht dieses Byte hier auf “03”.

Das gesamte Kommando lautet: 8A 01 00 00 03

Direkt nach dem es gesendet wurde wird die Karte das Kommando zunächst einmal bestätigen in dem sie einfach das Instruction-Byte zurückschickt.

Die Antwort lautet also zunächst: 01

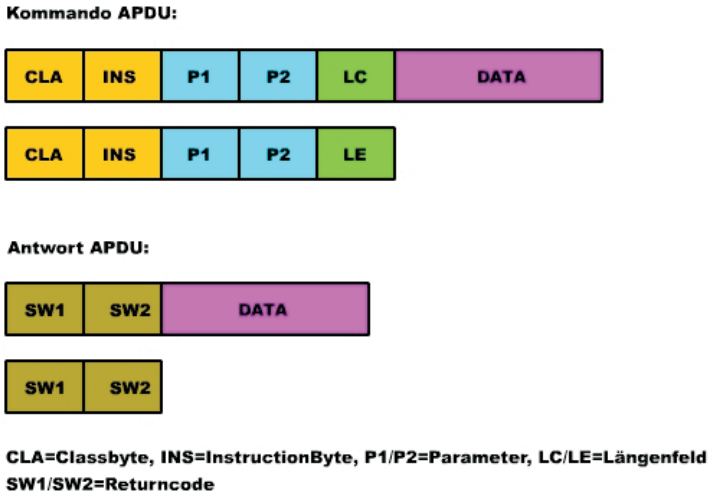
Es sei hier erwähnt das diese Bestätigung, wenn man auf einer höheren logischen Ebene arbeitet oft nicht sichtbar sind. z.B.: In Gscriptor sieht man von dem zurückgesendeten Instruction-Byte überhaupt nichts.

Jetzt könnte man theoretisch 03 Bytes Daten an die Karte senden, da das letzte Byte eigentlich nur angibt wie viel Daten man schicken möchte bzw. geschickt bekommen möchte.

Die Karte sendet also sofort danach: 31 30 30 90 00

“30 31 30“ ist ASCII und bedeutet “100” Es ist also die XCOS Version 1.0.0 installiert. Doch was bedeutet “90 00”? Nun, jede Antwort die von der Karte kommt wird mit einem sogenannten Return-Code abgeschlossen. “90 00” bedeutet so viel wie: “Alles ok!” Mit Return-Codes teilen SmartCards dem Terminal mit ob etwas schief gegangen ist.

In der Tat gibt es 4 verschiedene Fälle in denen Kommando-Antwort (sog. APDUs) Sequenzen auftreten können. Diese können hier natürlich nicht alle im Detail erleutert werden. Die folgende Grafik illustriert aber welche Kommando-Antwort-Sequenzen von XCOS verstanden werden.



Die ersten fünf Byte der Kommando APDU sind der Header, der Rest wird als Body bezeichnet.

Wenn der Befehl keine Daten, sondern nur einen Returncode liefert muss das LE Feld auf 00 gesetzt werden. Auch wenn dieses Byte hier null ist muß es auf jeden Fall gesendet werden da XCOS immer einen APDU-Header mit 5 Byte länge erwartet.

Jeder APDU-Header wird durch zurücksenden des INS-Bytes bestätigt.

Will man Daten senden setzt man das LC Feld entsprechend der Anzahl der Bytes im Datenteil.

Die Chipkartenbibel:

Natürlich ist das hier gezeigte zu wenig um die volle Tragweite der Thematik zu verstehen. Im Internet findet man zum Teil gute Publikationen und Diplomarbeiten zur Thematik. Wer sich allerdings ernsthaft mit der Thematik auseinandersetzen will, der sollte sein Wissen aus einer guten Quelle beziehen.

Ich empfehle: Rankl, Effing: Handbuch der Chipkarten. Das Handbuch der Chipkarten ist die meiner Ansicht nach vollständigste Referenz und sehr verständlich geschrieben. Dieses Buch machte im übrigen die Entwicklung der DeveloperCard und XCOS erst möglich.

Die XCOS Quelltexte mit entsprechender Dokumentation sowie Beispielskripte für Gscriptor sind unter http://www.runningserver.com zum Download verfügbar, ebenso wie das Platinenlayout der DeveloperCard. XCOS darf gem. GNU-GPL v2.0 verwendet werden. Die Platinenlayouts stehen unter Creativ-Commnos-Licence zur Verfügung.

Wichtige Daten zu ihrer Karte:

Daten:
PIN: 01 02 03 04 (Hexadezimal!)
ATR: 3B 7F 01 00 FE 58 43 4F 53 76 32 30 30 28 63 29 50 46 42 4D

Befehle:
8A 02 00 00 08 ==> Liefert CHIP ID
8A 01 00 00 03 ==> Liefert Versionsnummer des OS (ASCII)
8C 03 00 00 01 ==> PIN Zähler abfragen
8C 04 00 00 01 ==> PIN Status abfragen
8C 01 00 00 04 01 02 03 04 ==> Pin verifizieren (01 02 03 04)
8C 02 00 00 04 AA BB CC DD ==> Pin ändern (in AA BB CC DD)
8C 05 00 00 00 ==> Pin verifikation zurückziehen
8E 01 00 07 04 ==> 4 Byte aus dem EEPROM lesen, Page 0, Start bei Byte 7
8E 02 00 07 04 AB CD EF 23 ==> Daten (AB CD EF 23) ab Byte 7, Page 0 schreiben
8E 03 00 00 FF ==> FF Byte aus dem Flash-ROM lesen, Start bei Byte 0, Page 0

ACHTUNG: Bei dreimaliger falscher PIN Eingabe zerstört sich die Karte automatisch. Das Betriebsystem muss dann neu geflasht werden.

Tips und Tricks:

Auf der Karte befindet sich ein AVR-Microncontroler. Auf der Webseite http://www.mikrocontroller.net/ ist ein sehr anschauliches Tutorial zur AVR Programmierung zu finden, dieses sollten Sie sich durchlesen, sofern Sie mit der Materie noch keine Erfahrung haben. Das Tutorial beschreibt ebenfalls den Bau eines einfachen Programmiergerätes, welches Sie zwingend benötigen wenn Sie die Karte selbst flashen wollen.

Sollte ihnen der Microcontroller einmal kaputt gehen müssen Sie bei einem Neukauf darauf achten das der Controller so eingestellt ist das er externe Taktquellen akzeptiert. Wie das geht steht im Datenblatt. Die Konfigurationsbits können mit dem Programm PonyProg komfortabel gesetzt werden.

Vermeiden Sie jede Berührung der Kontakte der DeveloperCard, da die Kontakte aus blankem Kupfer bestehen oxidieren sie sehr leicht. Wenn Sie geschickt sind können Sie die Kontakte mit etwas Lötzinn und Entlötlitze verzinnen.

Wenn Sie ein Kartenterminal aus Metall verwenden kann es zu Kurzschlüssen kommen, isolieren Sie deshalb die Leiterbahnen auf der Oberseite der Karte mit einem Klebestreifen.