Mikroelektronik

Starter Kit für den C167

Christian Perschl

Ende vorigen Jahres sind sie überall aufgetaucht: Die Mikrocontroller-Starterkits von Siemens. Wozu sie gut sind, was dabei ist und wie man mit ihnen am besten zurecht kommt, soll in diesem Artikel behandelt werden.

1 Was bringt der Starter Kit?

Der Starter Kit ist nicht nur eine äußerst günstige Möglichkeit (ca. ATS 2000,- excl. MwSt) , 8 Bit und 16 Bit Mikrocontroller kennenzulernen, sondern auch mit Hilfe fertiger Hardware eine beliebige Applikation im Bereich Messen / Steuern / Regeln kostengünstig zu realisieren. Auch für Applikationen mit mehreren Boards, welche mittels CAN-Bus kommunizieren, sind Starter Kits sehr gut geeignet.

Es werden zu den verschiedensten Mikrocontrollertypen Starterkits angeboten: Angefangen bei den 8-Bit Controllern (der bekannten 8051er Familie) gibt es den C504-Starter-Kit (entspricht dem Typ 80C32 + PWM-Einheit), C505C (zusätzlich mit startklarem CAN-Bus), C515 (kompatibel zum weitverbreiteten 80C535), C541USB (es gibt endlich Hardware zum Universal Serial Bus ;-)).

Leistungsfähiger, moderner und daher für viele Fälle auch interessanter sind die 16 bittigen Bausteine, die 80C166 Familie. Hier werden für den C161, den C163, C164CI und dem C167CR Starter Kits angeboten. Auf den letzteren, welcher der am besten bestückte und leistungsfähigste Controller der Familie ist, wollen wir näher eingehen.

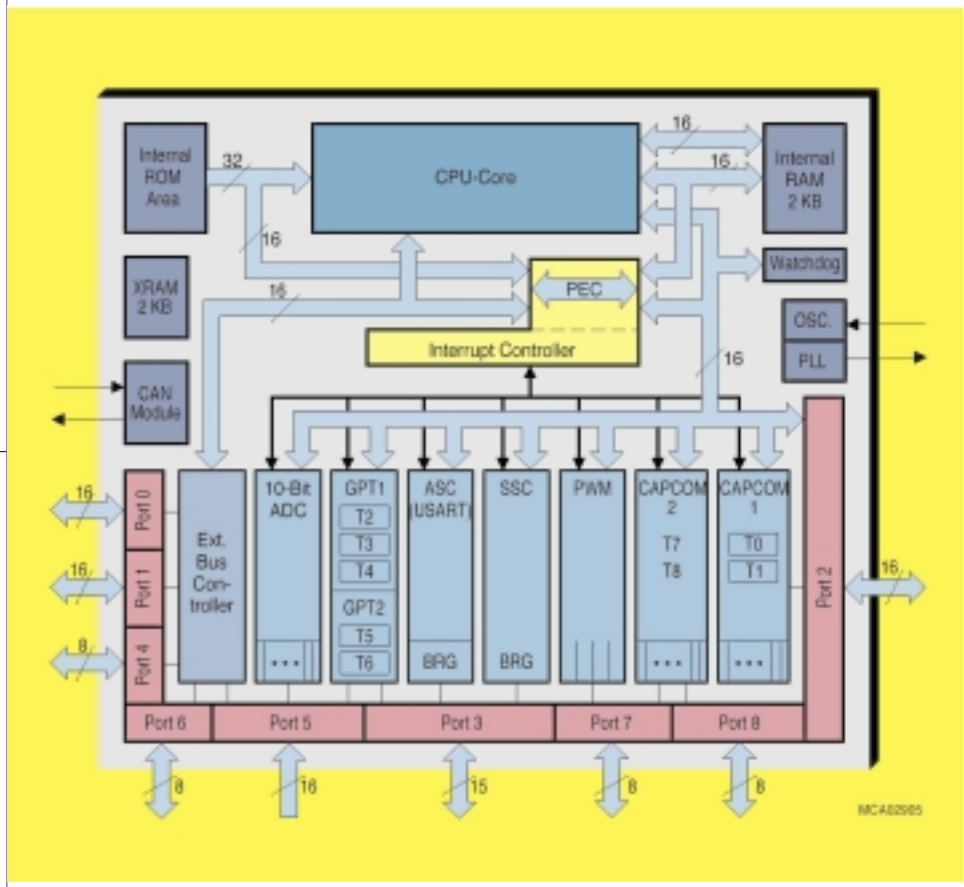
2 Zum C167

Nur ganz kurz, weil vielen ohnehin bekannt:

Der C167 ist ein von Siemens von Grund auf neu designter 16 Bit Mikrocontroller. Er arbeitet mit 80ns/100ns Befehlszykluszeit (aufgrund der Risc/Pipeline-Architektur wird fast jeder Befehl in dieser Zeit ausgeführt) kann bis zu 16 MB Speicher adressieren, beinhaltet als internen Speicher 4 Kbyte Ram (2 kB XRAM, 2kB dual ported). Mittels Bootstrap-Loader kann der Baustein "nackt" hochfahren, d.h. das Programm über die serielle Schnittstelle empfangen.

Das Interrupt System kommt fast einem Betriebssystem gleich: 16 verschiedene Interrupt-Prioritätsebenen, insgesamt 56 Interruptquellen. Zusätzliche 8 PEC (Peripheral Event Control) Kanäle erledigen bei Ereignissen beliebige Speichertransfers so nebenbei und ohne Rechenzeitverschwendung.

Interessant auch die Peripherie: 10 Timer/Counter mit verschiedensten Betriebsarten, Capture/Compare Einheit mit 16 Kanälen, 16 externe Interrupts, zusätzlich PWM Einheit mit 4 Kanälen und 50 ns



e9327470@stud1.tuwien.ac.at

Auflösung, 10 Bit Analog/Digital Wandler mit 10 us Wandlungszeit und 16 gemultiplexten Eingängen. Auch nicht ganz zu verachten: 111 I/O Portleitungen, frei als Eingang, Ausgang oder Open Drain konfigurierbar. Was soll man da wohl dranhängen ?

Er bietet auch einiges an Schnittstellen: die übliche asynchrone Schnittstelle mit bis zu 625 kBaud, eine sehr flexible synchrone Schnittstelle mit bis zu 5 MBaud und vor allem der mächtige CAN-Bus. Sollte dies nicht reichen, so können nebenbei z.B. mit den Timern und PEC-Transfers locker noch einige weitere Schnittstellenprotokolle emuliert werden.

Insgesamt ein flotter, dennoch unproblematischer Baustein mit mächtiger Peripherie, der da auf dem Starterkit sitzt.

3 Hardware

Der Starter Kit C167 beinhaltet neben dem Mikrocontroller C167 256 KByte Flash-Rom und 64 KByte RAM. Beide Speicher können auf bis zu jeweils 2 MByte erweitert werden. Im allgemeinen wird man wohl mit der Standardbestückung auskommen.

Weiters befinden sich auf der Platine Anschlüsse für die asynchrone serielle Schnittstelle (das Kabel zum Computeranschluß ist ebenfalls im Paket inkludiert) und für den CAN-Bus.

Ein Spannungsregler gewährleistet, daß eine nicht stabilisierte Eingangsspannung (8V-12V) auf 5 V stabilisiert wird.

Eine 152-polige Steckerleiste führt alle Anschlüsse des Mikrocontrollers heraus, wodurch eine einfache Erweiterung der Hardware möglich ist. Auch ein multifunktionaler Einsatz des Starter Kits ist dadurch erreichbar.

4 Dokumentation und Software

Im Paket befindet sich eine Reihe von papierener Dokumentation: In "User´s Manual", "Data Sheet" und "Instruction Set" des C167 sind wohl alle Informationen, die zur Programmierung des C167 erforderlich sind, enthalten. Außerdem ist eine Dokumentation der Hardware (inklusive Schaltplan) und eine Kurzanleitung zur Installation enthalten.

Eine CD mit Programmen und Beschreibungen aller Starter Kits ist dem Paket beigelegt ("Semiconductor Group - Microcontroller Starter Kits"). Das wichtigste Programm auf dieser CD ist wohl der Flash-Programmer FLASHT.EXE, welche vom Board-Hersteller PHYTEC zur Verfügung gestellt wurde. Damit ist es möglich, ein compiliertes Programm (in Form eines Intel Hex-Files) in das Flash zu schreiben und zu starten.

Daneben befinden sich eine Reihe von "bunt zusammengewürfelten" Dokumentationen, Beispielprogrammen, Tools, usw. auf der CD. Für Fortgeschrittene nicht ganz uninteressant: Application Notes zu den Themen CAN, Bootstrap, Flashprogrammierung u.ä. inklusive Programmgerüsten und -beispielen.

Nicht ganz zu verschweigen sind Demo-Versionen von Compilern und Assemblern. Diese Pakete sind zwar eingeschränkt, kleinere Programme können jedoch damit übersetzt werden. Wie stark diese Versionen eingeschränkt sind, wird im nächsten Kapitel beschrieben.

Eine weitere CD im Starter Kit enthält alle Datenblätter von Siemens Halbleitern ("Semiconductor-Technical Product Information").

Alle Informationen zu den Starter Kits, Data Manuals und Application Notes sind übrigens auch im Internet zu finden:

Mikrocontroller:
http://www.siemens.de/Mikrocontroller/

Starter Kits:
http://w2.siemens.de/
Semiconductor/products/ICs/34/index2.htm

5 Compiler

Nun würden wir gerne eigene Programme schreiben. Die Entscheidung, in welcher Sprache fällt ziemlich leicht: Es gibt mehrere C-Compiler (Ansi C und C++), und Assembler ist heute wohl nur noch für Kleinst-Applikationen sinnvoll, einerseits wegen des allgemeinen Programmieraufwandes, andererseits wegen Fehlersuche und Programmwartung.

Im wesentlichen sind 2 C-Compiler für den C167 zu empfehlen: Tasking und Keil. Dabei kann in einer Grobabschätzung so geurteilt werden: Tasking in der aktuellen Version beherrscht sowohl C++ alsauch Ansi C, wogegen Keil ausschließlich Ansi C übersetzt. Dafür hat Keil die wesentlich ausgereiftere Entwicklungsumgebung und mit dScope einen recht netten Simulator bzw. Debugger. Von Tasking gibt es eigentlich bislang überhaupt keine eigene IDE, die Holländer setzen auf die Universal-Umgebung Codewright, welche i.a. nicht Bestandteil des Compilerpaketes ist. Die Installation der Tasking-EDE hat sich als nicht immer problemlos herausgestellt. Der Debugger Crossview liegt dem Tasking-Paket ebenfalls bei.

Wer also CPP-Applikationen schreibt, wird sich für Tasking entscheiden, wer eine nette Windows-Oberfläche und einen ausgereiften Simulator/Debugger möchte, greift lieber zum Keil-Compiler.

Tja, leider haben die beiden Compiler einen weitaus größeren Haken: Sie sind nicht ganz billig. Im Gegensatz zu Standard-Compilern (z.B. Microsoft und Borland) sind sie bei weitem nicht so verbreitet, und der Entwicklungsaufwand für einen Compiler ist recht hoch. Man muß schon ziemlich in die Tasche greifen (ca. ATS 45.000), und da hört sich für viele der Spaß auf.

Nun, wem nützt der Starter Kit ohne Compiler ?

6 Demoversionen

Gott sei Dank gibt es Demoversionen von C-Compilern und Assemblern.

Auf der Starter-Kit CDROM sind Compiler (inklusive Assembler) von 4 Herstellern enthalten: Keil, Tasking, High-Tech (Gnu-Compiler) und Amit. Zusätzlich befindet sich auf der CD ein Assembler der Firma Ertec.

Die auf der CD enthaltenen Compiler von Amit und High-Tech sind offenbar ältere Versionen. Sie können vom Funktionsumfang her den beiden anderen Compilern nicht das Wasser reichen: nur wenige ANSI-C Bibliotheksfunktionen, keine vorgefertigten Startup-Codes zur uC-Initialisierung und eine etwas umständliche Installation. Es war zwar möglich, einfache Programme zu compilieren, am Starter Kit selbst funktoniert hat allerdings keines der Programme. Dies liegt wohl am fehlenden Startup Code, denn spätestens der Linker / Locater sollte wissen, wo welcher Speicher liegt.

Die beiden anderen Produkte (Tasking und Keil) sind wesentlich ausgereifter und haben einen entsprechend größeren Funktionsumfang.

Bezüglich den Einschränkungen der Demoversionen gehen Tasking und Keil grundsätzlich verschiedene Wege:

Keil schränkt den Compiler und den Assembler überhaupt nicht ein, erst der Linker entscheidet, ob ein downloadbares Programm generiert wird oder nicht. Die Limitation ist einfach: maximal 4 kByte Gesamtcode (nach dem Linken/Locaten) können generiert werden, und hier führt kaum ein Weg vorbei. Leider sind in diesen 4 kByte auch Konstante drinnen, da kann es bei größeren Konstanten (z.B. Strings) ziemlich eng werden. Naja, aber hier gibt es sicher Möglichkeiten....

Dagegen ist die Größe des Datensegmentes überhaupt nicht eingeschränkt. Es können im Prinzip beliebig viele (beim

Starter Kit je nach Speicherausrüstung bis zu 2 MB) Daten angesammelt werden.

Die Entwicklungsumgebung uVision und der Simulator dScope sind interessanterweise auf 8 kB Code beschränkt.

Das Tasking-Demo-Paket ist dagegen bei vielen Punkten eingeschränkt, am meisten beim Compiler: maximal 2500 Tokens (Variablennamen, Operatoren, Klammern, Schlüsselwörter...), maximal 500 Symbole (Variablen, Funktionen), keine Floating Point Operationen, ausschließlich das Speichermodell `SMALL`.

Ein Makroassembler ist überhaupt nicht im Paket, was die Übersetzung des Startup-Codes und damit eigentlich ein Laufen des Programmes am Board unmöglich macht. (Ich habe daher dem Artikel eine von Makroassembler nach Assembler übersetzte Startup-Datei beigelegt.)

Auch der Linker ist eingeschränkt: max. 60 Sections, max. 1000 Symbole, max. 13 Objektdateien und - jetzt kommts - max. 16 kByte an Daten und Code.

Die Entwicklungsumgebung Codwright bzw. EDE sind lächerlich eingeschränkt: "only very small programs", die EDE ist zum Erscheinungstermin dieses Artikels bereits abgelaufen. Da hilft wohl nur die Uhr zurückstellen ;-)

Klarerweise macht es wenig Sinn, mit viel Aufwand und Hirnschmalz zu versuchen, die Einschränkungen dieser Compilerpakete zu umgehen, die Programme sind vornehmlich zum Testen gedacht, und vernünftig arbeiten läßt sich nur mit den Vollversionen.

Trotzdem habe ich den Versuch unternommen, ein paar Grenzen herauszufinden und weiters ein "typisches" Anwendungsprogramm zu übersetzen und auszuprobieren, wie relevant die genannten Einschränkungen wirklich sind und wo sie zum Tragen kommen.

Der erste Test war ein "Hello World"-Programm, welches beim Keil auf Anhieb übersetzt wurde, beim Tasking prompt scheiterte. Dies liegt an der Einschränkung des Tasking-Compilers auf 500 Symbole. Durch Inkludieren des Headers reg167.h, in welchem alle SFRs definiert sind, wird diese Grenze bereits überschritten. Es empfiehlt sich daher, diesen Header gar nicht zu inkludieren und nur die benötigten SFRs wie in der Source zu definieren. Danach hat "Hello World" natürlich funktioniert.

Als nächstes sollten verschiedenste Grenzen erfahren werden. Die folgende Aufstellung soll über diese Tests ein biβchen Überblick verschaffen, die Werte sind aber natürlich mit Vorsicht zu genießen:

Der dritte und am wohl am meisten aussagekräftige Test war eine "typische" Applikation: Über die serielle Schnittstelle wird mit dem PC kommuniziert. Über ein Terminal können verschiedene Peripherie-Einheiten wie Timer, A/D Wandler, externe Interrupts und synchrone serielle Schnittstelle initialisiert und gestartet werden. Das Programm beinhaltet in einigermaβen ausgewogenem Verhältnis Konstanten, etliche Funktionen und entsprechende Interrupt-Handler. Insgesamt umfaβt das Programm ca. 350 Programmzeilen.

Tasking hat die Übersetzung des Programmes gerade noch geschafft, ein weiterer Befehl und das Programm kippt in die Nichtübersetzbarkeit.

Der Keil-Compiler ist an der 4kB-Grenze gescheitert, allerdings nur sehr knapp: Die Reduktion der Konstanten um ca. 100 Bytes genügte, damit auch Keil das Programm übersetzt. Die beiden Demo-Compiler dürften trotz der unterschiedlichen Restriktionen ziemlich gleich viel übersetzen. Ob das ein Zufall ist ?

Fazit

Mit der Keil-Demo lassen sich kleinere Applikationen übersetzen, vor allem bei Verwendung von viel Datenspeicher ist er dem Tasking vorzuziehen. Auch Entwicklungsumgebung und Simulator sind ein Argument für Keil. Ein Lehrbetrieb mit der Demoversion ist bis zu einem gewissen Grad durchaus möglich, man muβ sich halt immer die 4k-Grenze vor Augen halten.

Tasking kann i.a. eine Spur mehr Code übersetzen, hat aber nur die weniger brauchbare und ohnehin bereits abgelaufene Entwicklungsumgebung. Außerdem bedarf es einer Reihe von Tricks, um ein Programm übersetzen zu können (siehe Startup-Code, Weglassen der SFR-Definitionen). Auch Floating Point ist für den Demo-Tasking ein Fremdwort.

Bis auf die zusätzlichen Datentypen (`Bit`, `SFR`), Steuerwörter (`interrupt`, `idata`, `far`, `huge`...) und Intrinsic-Funktionen (spezielle Befehle des Mikrocontrollers) sind die Quellcodes der beiden Compiler identisch. Daher kann man ohne viel Aufwand die Sourcen übertragen und einfach ausprobieren, welcher Compiler ein Programm übersetzt und welcher nicht.

Programm	Beschreibung	Keil	Tasking
maxdata	Wie viele Bytes Daten können verwendet werden?	unbegrenzt	16 kB
maxkonst	Wie viele Bytes Konstante können verwendet werden?	4kB	16 kB
maxfunc	Wie viele einfache Funktionen in einer Source?	400	100
maxsfr	Wie viele Special Function Register in einer Source?	alle	450
maxcode	Wie viele Speicherzuweisungen in einer Source?	650	300
maxkonstr	Wie viele Konstrukte (IF/ELSE) bzw. (FOR-Schleifen)?	150/130	65/55
maxvar	Wie viele Variablen können maximal definiert werden?	unbegrenzt	450

7 Inbetriebnahme

Hardwareinstallation

Nach dem Auspacken breitet sich der oben beschriebene Inhalt des üppig gefüllten Starter Kits auf dem Tisch aus. Um alle Teile und Bücher wieder ordnungsgemäβ in das Paket zuückzuverfrachten, gehört wohl ziemlich viel Schlichtungssinn dazu, ich habe es jedenfalls nicht mehr geschafft.

Das μC-Board als das Kernstück des Starter Kits ist rasch erkannt, auch das beiliegende Kabel für die serielle Schnittstelle. Zuerst sollte man das Kabel an eine der beiden RS-232 Schnittstellen am PC anschließen, dann erspart man sich die Entscheidung beim Board, da dieses 2 Buchsen enthält (RS232 9-polig weiblich und CAN-Bus 9-polig männlich).

Damit das Board auch Saft bekommt, ist noch ein (ungeregeltes) Netzteil mit Gleichspannung von 8-12V mit 500 mA erforderlich. Dieses ist nicht im Starter Kit enthalten, gibt´s aber in jedem Elektronik-Geschäft (ca. ATS 100,- bis 150,-)

Nach Anlegen der Versorgungsspannung sollte die rote Versorgungs-LED am Board erleuchten. Tut sie dies nicht, dann ist guter Rat teuer und entweder der Starter Kit oder das Netzteil defekt.

Softwareinstallation

Damit wir einmal die Funktionstüchtigkeit des Boards und die Kommunikation mit dem PC überprüfen, ist der erste Schritt das Installieren der Flash-Programmierroutine. Diese befindet sich auf der Starter Kit CD-Rom im Verzeichnis

```
\CDROM\3RDTOOLS\PHYTEC\FLASH166\FLASHT\167
```

und braucht lediglich auf die Festplatte kopiert werden. Ein geeignetes Beispielprogramm mit dem Namen HELLO167.H86 befindet sich ebenfalls auf der CD-ROM unter

```
\CDROM\3RDTOOLS\PHYTEC\FLASH166\FLASHT\
EXAMPLE\hello167.h86
```


e9327470@stud1.tuwien.ac.at

Vor dem Start des Programmes FLASHT sollte der Reset-Taster am Starter-Kit Board gedrückt werden (Er ist schwer zu verfehlen, es ist der einzige Taster). Dabei ist zusätzlich darauf zu achten, daß der Jumper JP2 (roter Jumper, defaultmäßig on) gesetzt ist. Dadurch wird der C167 in den Bootstrap Loader Mode versetzt, welcher zur Flash-Programmierung erforderlich ist.

Das Programm FLASHT.EXE

Im Programm FLASHT erscheint nach dem Download der Flash-Utilities das Menü. Sollte der Ladevorgang bzw. das Menü nicht erscheinen, dann ist entweder ein Fehler in der Verbindung zwischen Board und PC, oder der C167 ist nicht im Bootstrap-Mode. Im zweiten Fall reicht es, wieder auszusteigen (***F1***), den Reset nochmals zu drücken und das Programm wieder zu starten.

Im Menü ist es am einfachsten, mit dem Befehl ***7) Erase, Load and Software Reset*** das Programm in das Flash zu schreiben. Nach dem Löschen aller Flash-Bänke wird nach der Programmdatei gefragt. Dazu muß die Taste ***F2*** gedrückt und der Dateiname eingegeben werden.

Nachdem das Programm geladen wurde, wird der C167 zurückgesetzt und das Programm gestartet. Das Programm gibt auf der seriellen Schnittstelle "Hello World" aus. Diese Zeichenfolge sollte dann auch am Bildschirm erscheinen.

Danach kann mit ***F1*** das Programm beendet werden.

Um das Programm im Starter Kit wieder zu starten, muß - es befindet sich ja bereits im Flash - lediglich der Resettaster kurz gedrückt werden. Hier darf sich der C167 keinesfalls im Bootstrap Loader Mode befinden, d.h. der rote Jumper JP2 muß offen sein. Erst wenn ein neues Programm geladen werden soll, muß der Jumper wieder gesetzt sein.

8 Starter Kit zum Starter Kit

Am schwierigsten gestaltet sich es, ein lauffähiges Programm für einen der beiden Compiler zu schreiben. Es werden im Vergleich zu einem PC-Programm zusätzliche Teile benötigt: Startup-Code zur Initialisierung, Locateranweisungen, I/O-Routinen...

Für einen Einsteiger ist es offen gesagt unmöglich, ohne allzu großem Zeitaufwand etwas Vernünftiges und Lauffähiges zustandezubringen. Nun ja, es gibt zwar Beispielprogramme auf der CD, diese sind jedoch nicht auf den Starter Kit zugeschnitten, Tasking gibt nicht einmal eine Startup Datei dazu.

Deshalb stelle ich Programmgerüste - sowohl für Tasking als auch für Keil - zur Verfügung, bei denen alle für die Hardware erforderlichen Einstellungen bereits vorgenommen sind und im Prinzip nur mehr C-Code in der "Hauptdatei" eingesetzt werden muß.

Keine Hürde, aber als oft lästig stellt sich das Flash-Tool heraus: Man kann das Programm nicht per Parameter aufrufen, sondern muß in mehreren Schritten zuerst das Flash löschen, die Datei zum Programmieren angeben ....

Als Alternative habe ich den in ***[2]*** beschriebenen Hex-Loader umgeschrieben. Mit seiner Hilfe ist es möglich, per Parameter und damit auch per Stapelverarbeitung ein Programm in das Flash zu schreiben. Außerdem ist der Lösch- und Programmiervorgang wesentlich flotter, da der Hexloader das komplette Flash löscht. Das Programm heißt ***CPFLASH*** und ist ebenfalls beigelegt.

Für Tüftler lege ich noch das Testprogramm "typische" Applikation sowie die "max"-Programme, welche für den Demo-Compiler-Test verwendet wurden, bei. Außerdem können sie als Beispiele für die beiden Compiler dienen.

9 Resumée

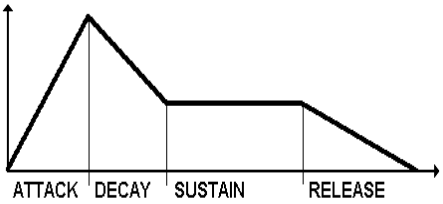
Ich habe mir bereits zwei Starter Kits zugelegt und losgelegt. Und ich merke, daß Dank Starter Kit der Phantasie keine Grenzen gesetzt sind: MIDI-Schnittstelle, Eprombrenner, Lichterorgel, Metronom oder Plotteransteuerung. Oder einfach nur zum Kennenlernen.

10 Programme

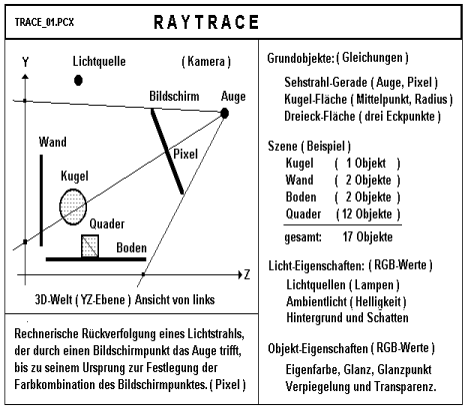
Unmittelbar bei der Web-Version dieses Artikels finden Sie im Verzeichnis prg die folgenden Programme jeweils in einem eigenen Verzeichnis:

```
cpflash demotest example typapp
```

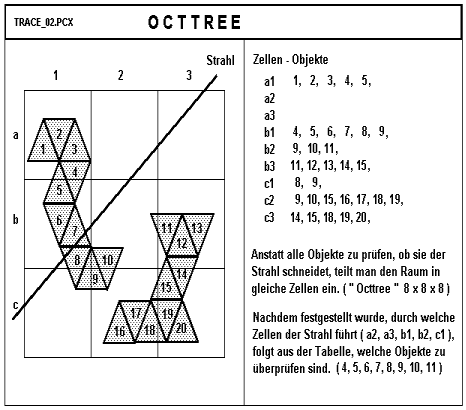
sound_3: Hüllkurve



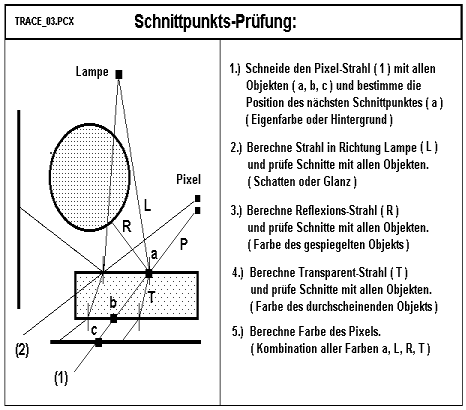
trace_01: Raytrace



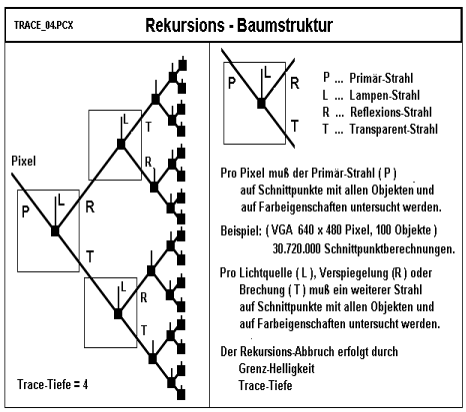
trace_02: OCTTREE



trace_03: Schnittpunktsprüfung



trace_04: Rekursions-Baumstruktur



Mikroelektronik