

# OPL Workshop 1a - Einstieg für den Psion S3a/c

Ulrich Krinzner, 6/1998

## - Beispiele, Tips, Ratschläge -

*Dieser Beitrag richtet sich an Psionisten, die sich mit der OPL-Programmierung auf dem S3a/c auseinandersetzen wollen. Wir werden über die Programmierung allgemein reden und haben Tips und Beispiele für die praktische Seite.*

*Nachdem einmal der Wunsch gewachsen ist, sich mit OPL auseinanderzusetzen, steht jeder Einsteiger vor der Frage nach dem WIE? Das betrifft sowohl das Programmieren selber als auch das zugehörige Werkzeug, also einen wie auch immer gearteten Rechner mit geeigneter Software. Beginnen wir mit letzterem.*

*Sie verfügen über eine ganze Palette an Möglichkeiten, jedoch hängt von Ihren persönlichen Umständen ab, welche Sie davon auch nutzen können. Wir stellen Ihnen die einzelnen Varianten mit Vor- und Nachteilen vor. Entscheiden müssen Sie - und Ihr Geldbeutel - allerdings selbst ...*

## Ihr Handwerkszeug

### Programmieren mit dem Psion

Das ist die einfachste und kosteneffektivste Art, die Dinge beim Schopf zu packen. Akzeptieren Sie unsere Empfehlung: besorgen Sie sich im Fachhandel das passende Netzteil. Ihr Portemonnaie wird es Ihnen danken, der Batteriehändler wahrscheinlich weniger.

**Die Vorteile:** alle benötigten Komponenten sind in Ihrem Psion bereits vereint: der Editor, das Übersetzungsmodul und die Testmöglichkeit des gerade bearbeiteten Programmes.

**Die Nachteile:** Was im mobilen Betrieb von Vorteil ist, stellt sich hier als hinderlich heraus: Tastatur und Bildschirm sind für ein Programmier-Marathon etwas zu klein. Der Editor ist auf 32 kB beschränkt - zuwenig für umfangreichere Programme.

### Programmieren mit dem Psion-Emulator am PC

Psion hat dereinst für eigene Zwecke einen Emulator entwickelt. Er bildet den Psion 3a auf dem PC unter DOS weitgehend vollständig nach (1).

**Die Vorteile:** Geeignet konfiguriert, hat man mit dem Emulator den ganzen Bildschirm zur Verfügung und kann mit der großen Tastatur arbeiten. Es kommt tatsächlich ein Psion-feeling auf, die Nachbildung ist gut gelungen.

**Die Nachteile:** Die Einschränkung hinsichtlich der Editorgröße bleibt erhalten. Größere Programme erfordern eine ungewollte Aufspaltung in Teilmodule. Die Tastatur will nicht immer so, wie man sich das vorstellt; einige Zeichen lassen sich mit einer deutschen Tastatur garnicht erzeugen - der interne Treiber steht auf englischem Zeichensatz. Die amerikanische Programmversion ist auch nicht hilfreich. Allerdings kann man ein zusätzliches Programm in Anspruch nehmen, das per Makro die fehlenden Zeichen generiert: MACSYS von Tom Dolbiline (1a). Wem ein bißchen Umstand nichts ausmacht, der kann das Problem auch mit "holen" oder "kopieren und einfügen" angehen.

Nach Fertigstellung übertragen Sie Ihr Programm zum Test oder zur Benutzung auf den Psion. Dazu wird üblicherweise das spezielle serielle Link-Kabel verwendet. Das ist eine Investition von ca. 100 .. 150 DM, die man in allen Fällen tragen muß, in denen man die Programmentwicklung außerhalb des Psion durchführt.

## Programmieren am PC unter DOS

Wenn Sie lieber mit Ihrem vertrauten Texteditor arbeiten: tun Sie das. Sie können den Programmtext nach getaner Arbeit durch ein Übersetzungsprogramm (S3atran.exe) in ein lauffähiges Programm umwandeln. Den Test führen Sie nach Belieben aus: mit dem Emulator auf dem PC oder nach Übertragung der Dateien auf dem Psion.

"S3atran" ist Bestandteil des ODE-Kits (s.u.). Sie finden dieses Programm aber auch z.B. über das Psion-Forum bei CompuServe (2)

**Die Vorteile:** Sie arbeiten in Ihrer gewohnten Umgebung; eine Dateigrößenbegrenzung existiert nicht. Der Programmtest kann auf dem PC mit dem Emulator erfolgen, so daß keine Datenübertragung notwendig ist.

**Die Nachteile:** Die Unterstützung während des Programmierens ist denkbar gering und nicht OPL- bzw. Psion-bezogen.

## Programmieren am PC unter Windows 3.x/95

Auf den Einsatz eines Standardeditors wollen wir hier nicht weiter eingehen, denn das bringt kaum Vorteile gegenüber der DOS-Variante.

Die Firma Psion hat speziell für OSL eine Programmierumgebung unter Windows entwickelt: ODE (OSL Development Environment)

**Vorteile:** Sie haben in dem speziellen Editor alle Windows-Bequemlichkeiten zur Verfügung: es gibt eine Hilfedatei, die OPL-Befehle werden als Referenz bereitgestellt, "kopieren und einfügen" machen das Leben leicht. Weiter im Lieferumfang: der Übersetzer mit Preprozessor, OPLLINT (erweiterte Fehlersuche), Wandler für Bild-(PIC/PCX) und Sounddateien (WAV/WVE) und diverse Programm-Muster und -Rahmen. Die Link-Verbindung wird direkt aus ODE angesprochen.

**Nachteile:** Ohne Englischkenntnisse haben Sie Probleme, die bereitgestellten Informationen zu nutzen. Weiterhin: der Test des erstellten Programmes kann nicht unter Windows erfolgen (der Emulator arbeitet nicht sauber oder garnicht unter Windows). Die Dateien müssen immer erst per Link-Kabel auf den Psion übertragen werden.

ODE erhält man als Bestandteil des OPL SDK (Software Development Kit) in der Professional-Version z.B. vom PSIologic Versand (3). Neben dieser Software enthält das Kit sehr ausführliche Handbücher zu OPL und zu den Dingen drumherum. Ausführliche Informationen holt man sich am besten aus dem Internet direkt von Psion PLC (4), oder man stöbert bei entsprechenden anderen Psion-Anbietern.

Die reine ODE-Software wird u.a. von Widget/GB (5) angeboten. Man bezahlt allerdings die höhere Verbrauchssteuer und der Kurs des Pfunds ist auch nicht mehr, was er mal war.

Eine ganz private Variante des Autors sieht so aus:

Editieren des Programmes unter Windows 95 mittels ODE, dort auch übersetzen lassen und mit OPLLINT auf Fehler prüfen. Anstelle nun die Dateien mit langsamen 19 kbits/sec auf den Psion zu übertragen, werden sie auf eine 3,5 Zoll Diskette überspielt. Diese wiederum findet Platz in einem Laptop oder Zweitcomputer, den man sich für die Arbeiten von Freunden borgt. Auf diesem Gerät läuft dann schon der Psion-Emulator warm ...

Der Vollständigkeit halber: Sie haben für die Programmentwicklung weitere Möglichkeiten, die dann aber jenseits von OPL liegen. Das sind:

- OVAL (Object-based Visual Application Language), eine Art Visual Basic, das auf die Belange des Workabout und S3c zugeschnitten wurde , und
- die Programmiersprache "C".

Unser kleiner Ausflug zu den Programmierwerkzeugen endet hier. Nun brennt die Frage nach dem "wie programmieren" auf den Nägeln. In kompakter Form ist das eigentlich garnicht zu beantworten, denn die Antworten füllen ganze Bibliotheken. Wir können Ihnen auch nicht die Mühe des eigenen Experimentierens abnehmen, das Nachschlagens im Handbuch müssen Sie auch selber bewerkstelligen. Aber wir haben ein paar Empfehlungen für den Anfang:

## Menüs und mehr

Sie können überall die Meinungen von Zeitgenossen darüber nachlesen, was man wie zu machen hat. Und Sie können beim Programmieren davon beliebig nach eigenem Geschmack nutzen. Tun Sie jedoch sich und den Nutzern Ihrer Programme etwas Gutes: stellen Sie einen zuverlässigen und erkennbaren Programmausstieg zur Verfügung. Nichts ist lästiger, als den "Abschaltknopf " nicht finden zu können.

Der einfachste Weg dazu ist die Nutzung der OPL-Befehle mINIT/mCARD/MENU. Wie ein damit erzeugtes Menü-System funktioniert, ist anschaulich an dem Beispiel im Programmierhandbuch (V.1, dt., 1993, S. 86) nachzuvollziehen. Der Vorteil: der Nutzer ist gewohnt, die Menütaste zu verwenden, und nun ist sie tatsächlich auch aktiviert.

Lassen Sie uns einige zugehörige Aspekte ergänzen.

### Gezielten Ausstieg ermöglichen

Das Menü braucht nicht viele Punkte zu enthalten. Es reicht, den Programmausstieg zu erleichtern. Sinnvollerweise definiert man dazu den Menüpunkt "Beenden" und ordnet ihm den Hotkey Psion-X zu. Der Hotkey ist die Möglichkeit, durch Halten der Psion-Taste und Drücken einer Buchstabentaste den Menüaufruf über die Menütaste zu umgehen. Bei Menüs ist das allgemein die schnellere Methode: ein einzelner Tastendruck reicht!

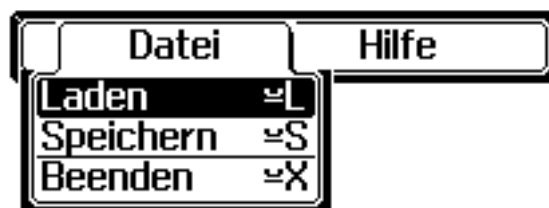


Abb.: Standardmenü , Psion-feeling

Damit das Psion-feeling erhalten bleibt, bezieht man sich bei der Auswahl der Buchstaben für die Standardfunktionen (Beenden, Laden, Speichern ...) am besten auf die "Norm", die der Psion in seinen eigenen Menüs mitbringt. Zu beachten ist die Auswahl der Landessprache. Aber auch bei deutscher Sprache sollte "Beenden" immer mit "x" (für "eXit") verknüpft sein, eben dem Standard. Beherzigt man diese einfache Regel, ist die Einarbeitung in ein neues Programm für den Anwender weitaus weniger aufwendig. Er wird es Ihnen danken - oder noch besser, er merkt es garnicht. Dann wird er Sie wenigstens auch nicht verdammen ...

Soweit die Theorie. Sie nutzt aber nichts, wenn Sie lediglich die Beispielfunktion oder eine vergleichbare andere abschreiben. Ihre Eigenschaften werden erst aktiviert, wenn sie überhaupt durchlaufen wird. Konstruieren Sie Ihr Programm deshalb so, daß es im entscheidenden Teil in eine Schleife eingebettet ist. In dieser Schleife muß die Tastatur dann ständig abfragt und ausgewertet werden. Da die meisten Programme sowieso die Tastatur abfragen müssen, ist das praktisch immer sinnvoll. Warum dann nicht gleich "richtig" machen?

### Tastaturabfrage - immer frisch

Um ständig erneuerte Informationen von der Tastatur zu erhalten sind die Befehle GET und GET\$ nicht besonders geeignet. An der entsprechenden Programmstelle wartet nämlich das Programm lediglich faul auf den nächsten Tastendruck und macht sonst gar nichts. Das folgende Programm bietet eine Alternative:

```
PROC quest:
  LOCAL k%(6), a%
  DO
    IF TESTEVENT
      GETEVENT k%()
      a%=k%(1)
      IF a%=&122      REM Menü-Taste
        REM rufe das Menü-Unterprogramm auf
      ELSEIF a% AND &200  REM Psion + Hotkey
        REM rufe das Psion-Hotkey-Unterpramm auf
      ELSEIF a%=291     REM Help-Taste
        REM rufe das Help-Unterprogramm auf (HelpOut: )
      ELSE
        REM prüfe noch anderes und reagiere
      ENDIF
    ENDIF
    REM führe noch andere Programmteile aus
  UNTIL 0
ENDP
```

Über TESTEVENT/GETEVENT informiert das Programmierhandbuch ausführlich. In aller Kürze: nur wenn überhaupt ein Ereignis dem Psion gemeldet wird, findet auch eine Auswertung statt. Durch geeignete Auswahl der Variablenwerte wird sie auf die Tastatur-Ereignisse eingeschränkt. Im Bedarfsfall wird entsprechend mit einem Unterprogramm reagiert. Ist kein Tastendruck auszuwerten, ist die Schleife in der Lage, andere Aufgaben auszuführen. Beispielsweise könnte eine Zeitanzeige aktualisiert werden.

Prinzipiell lassen auch die Befehle KEY und KEY\$ die Schleife weiterlaufen. Wir haben jedoch keine guten Erfahrungen damit gemacht. Besonders unzuverlässig war das Verhalten, wenn die Schleifeninhalte etwas länger ausfielen.

### HELP einbinden

Da wir gerade so schön dabei sind: Aktivieren Sie doch auch die spezielle Help-Taste des Psion, der Nutzer erwartet es nahezu. Die Helpkey-Abfrage ist in der obigen Programmschleife schon vorgesehen. Fehlt lediglich ein kleines Unterprogramm, das den anzuzeigenden Text darstellt. Hier hilft - OPL sei's gedankt - dINIT/dTEXT/DIALOG ganz schnell auf die Sprünge:

```
PROC HelpOut:
  dINIT "Hilfe für mein Programm"
  REM für einzeiligen Text, sonst ergänzen
  dTEXT "", "Erste Zeile des Hilfetextes, Zeilenlänge erproben!"
  ...
  DIALOG
ENDP
```



**Abb.: Hilfeanzeige, selbst gebaut**

Das ist zwar nicht die Erscheinungsform des Psion-Hilfebildschirms, erfüllt aber bei nur geringem Aufwand seinen Zweck ganz hervorragend.

## Menüs 'mal anders

Mit Hilfe der DIALOG-Konstruktion lassen sich auch Menüs gestalten. Ein Zusatz in der dTEXT-Zeile sorgt dafür, daß eine derartige Zeile durch die Pfeiltasten anwählbar wird. Versieht man dINIT noch mit einem Info-Text, ist das etwas andere Menü fertig :

```
PROC NewMenu:
  LOCAL a%
  dINIT "Menü Datei"
  dTEXT "", "Laden", $402
  dTEXT "", "Speichern", $602
  dTEXT "", "Beenden", $402
  a%=DIALOG
  IF a%=2
    ...
  ELSEIF a%=3
    ...
  ELSEIF a%=4
    ...
  ENDIF
ENDP
```



Abb.: Menü einfacher Art

Die Hexadezimalzahlen hinter der dINIT-Zeile bewirken die Selektierbarkeit, Textzentrierung und Abtrennung zwischen den Zeilen. Es muß nicht mit Hexadezimalzahlen gearbeitet werden - das dezimale Pendant funktioniert natürlich auch. Nach Eingewöhnung sind aber die "Hexzahlen" einfach übersichtlicher.

Je nachdem, welche Zeile während des Programmlaufes selektiert wurde, erhält a% einen bestimmten Wert. Dieser nun dient der Auswahl des entsprechenden Programmzweiges. Ein ESC liefert eine Null zurück und wird in diesem Beispiel daher nicht berücksichtigt. Die "eins" kann nicht vorkommen, sie gehört zur dINIT-Zeile.

Für manch einen mag es sich als Nachteil herausstellen, daß man bei dieser Form auf die Hotkeys verzichten muß. Aber man hat ja die Wahl.

Geht es um einfache Programme, macht man oftmals keine großen Klimmzüge für ein Menü .

In diesen Fällen ist es auch mal mit einem Druck auf die ESC-Taste getan. Nur sollte man das dem Anwender auch deutlich mitteilen! Das folgende kleine Umrechnungsprogramm für Temperaturen von Celsius zu Fahrenheit tut genau das:

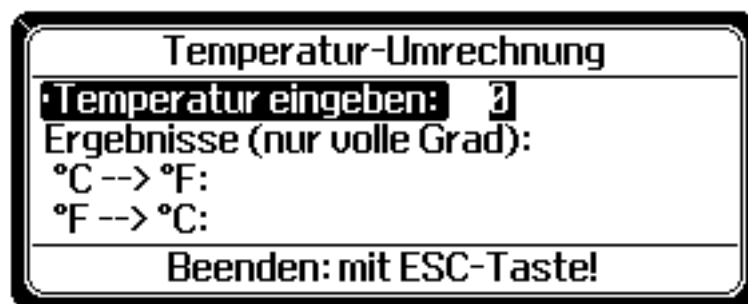
## Ein Temperaturkonverter

Wenn hier der Dialog mit der ESC-Taste abgebrochen wird, ist das Ergebnis von DIALOG wegen der Definition dieses OPL-Befehls "null", und der ELSE-Zweig wird durchlaufen - das Programm also beendet:

```

PROC tconv:
  REM Umrechnung von Temperatureinheiten; es gilt: °C=(°F-32)/1,8
  GLOBAL fz, c2f$(10), f2c$(10), c2f, f2c
  c2f$=" " : f2c$=" "
  DO
    dINIT "Temperatur-Umrechnung"
    dFLOAT fz,"Temperatur eingeben: ",-10000,10000
    dTEXT "", "Ergebnisse (nur volle Grad): "
    dTEXT "°C --> °F: ",c2f$,1
    dTEXT "° --> °C: ",f2c$,201
    dTEXT "", "Beenden: mit ESC-Taste!",2
    IF DIALOG
      c2f=32+(1.8*fz) : c2f$=FIX$(c2f,0,10)
      f2c=(fz-32)/1.8 : f2c$=FIX$(f2c,0,10)
    ELSE
      BEEP 3,333 : STOP REM Programmausgang
    ENDIF
  UNTIL 0
ENDP

```



Sehr einfaches Temperatur-Umrechnungsprogramm

Das ist in diesem Fall sogar ganz praktisch, da durch die Abfrage der Temperatur das Programm wie bei einem GET-Befehl jedesmal an der Stelle dFLOAT auf eine Eingabe wartet und gar nicht auf eine Menüanfrage reagieren könnte.

Nach diesem Beispiel können Sie auch beliebige andere Konverter aufbauen (km/miles ...).

Bevor wir Sie Ihren Experimenten überlassen, müssen wir noch den letzten Rat für heute loswerden: Kommentieren Sie Ihre Programme reichlich! Eine Binsenweisheit und im Moment lästig, ja, ja. Aber denken Sie an das nächste Mal und Ihre Freude darüber, wenn Sie das Rad nicht noch einmal erfinden müssen ...

Und nun viel Spaß beim Programmieren!

## Quellenverzeichnis

- (1) Psion S3a-Emulator: S3AEMU.ZIP, CompuServe: GO PSION, Lib.18
- (1a) Tom Dolbilin, MACSYS.ZIP v.2/1994, CompuServe: GO PSION, Lib. 18
- (2) S3ATRAN.EXE, CompuServe: GO PSION, Lib. 18
- (3) PSIologic , <http://www.psiologic.com>
- (4) Psion PLC, <http://www.psion.com>
- (5) Widget Software Ltd, <http://widget.co.uk>