UNIVERSITÄTSBIBLIOTHEK BRAUNSCHWEIG

FLEX

Eine Einführung in die Makrosprache des Systems ***allegro***

Dipl.-Math. B.Eversberg
Dipl.-Ing. C. Elsner

[Ein Einführung in die Makrosprache des Systems *allegro*]
Überarbeitete Auflage Mai/Juni 2007

0 Inhaltsverzeichnis

1 Was ist *FLEX* und was macht man damit?

Unter *FLEX* versteht man die Makro- oder auch Skriptsprache des *allegro*-Systems. Das bedeutet: mit *FLEX* kann man wiederkehrende Aktionen, wie beispielsweise Tastatureingaben und/oder Mausklicks automatisieren. Die entsprechenden Befehlsfolgen werden dann durch den *FLEX* so abgearbeitet, als hätte man die Befehle manuell eingegeben. Damit ein *FLEX* die gewünschte(n) Funktion(en) ausführt, benötigt er die entsprechenden Instruktionen, die *FLEX*-Befehle. *FLEX*-Befehle sind definierte Anweisungen, die dem System mitteilen, was getan werden soll. Das *allegro*-System stellt momentan etwa 100 *FLEX*-Befehle zur Verfügung (siehe Kapitel 9.1 und 9.2) und die Einsatzmöglichkeiten sind dementsprechend vielfältig. So sind beispielsweise nahezu alle Menüpunkte und Buttons von *allegro* auch durch *FLEX*-Befehle auslösbar.

Seit seiner ersten Einführung 1996 (als Befehlssprache für den avanti-Server) hat sich *FLEX* zum mächtigsten Werkzeug des allegro-Systems entwickelt. Inzwischen kann man umfangreiche Anwendungen damit entwickeln, so z.B. die Erwerbungs- und Ausleihfunktionen.

Merke: *Ein FLEX ist eine Abfolge von Anweisungen (FLEX-Befehlen), mit dem man wiederkehrende Abläufe wie Tastatureingaben und Mausklicks automatisieren kann.*

Jedoch bedeutet dies nicht, daß man als Benutzer des *allegro*-Systems für jeden Anwendungsfall seinen eigenen *FLEX* schreiben muß – *allegro* liefert selbstverständlich für die gebräuchlichsten Fälle fertige *FLEXe* (siehe Kapitel 9.4) mit. Ein großer Vorteil ist dabei, daß man solche *FLEX*e für eigene Zwecke anpassen kann, während fest eingebaute Programmfunktionen so etwas nicht ermöglichen würden. Früher mußten z.B. für die Ausleih- und Erwerbungsfunktionen eigene Programme in C geschrieben werden, in die dann kein Anwender eingreifen konnte!

Zunächst jedoch zurück zu den *FLEX*-Befehlen. Für den neuen Benutzer von *allegro* stellt sich an dieser Stelle selbstverständlich die Frage wie werden die *FLEX*-Befehle benutzt, wie gebe ich sie ein, und welche Befehle gibt es überhaupt?

2 Benutzung und Eingabe von *FLEX*-Befehlen

Erfahrene allegro Nutzer werden die Antworten schon wissen. Da diese Einführung aber auch - und nicht zuletzt - für die neuen und/oder unerfahrenen *allegro*-Nutzer bestimmt ist, soll darauf zuerst eingegangen werden. Richten wir unser Augenmerk zunächst einmal auf die Benutzung und die Eingabe von *FLEX*-Befehlen. Nach dem Starten von *allegro*, erkennt man neben dem Menü und einigen Schaltflächen auch Bereiche, die offenbar für die Benutzung von *allegro* wichtig sind, die so genannten Felder (siehe Abbildung 1). Wichtig für die ersten Schritte mit *FLEX-Befehlen* ist zunächst das Schreibfeld. Hier können dem *allegro*-System die *FLEX*-Befehle mitgeteilt werden, die ausgeführt werden sollen. Dazu schreibt man zunächst ein '***x***', um *allegro* mitzuteilen, daß statt einer Dateneingabe ein *FLEX*-Befehl folgt. Dem *x* und einem Leerzeichen folgen der oder die *FLEX*-Befehl(e) mit den eventuell dazugehörigen Parametern.

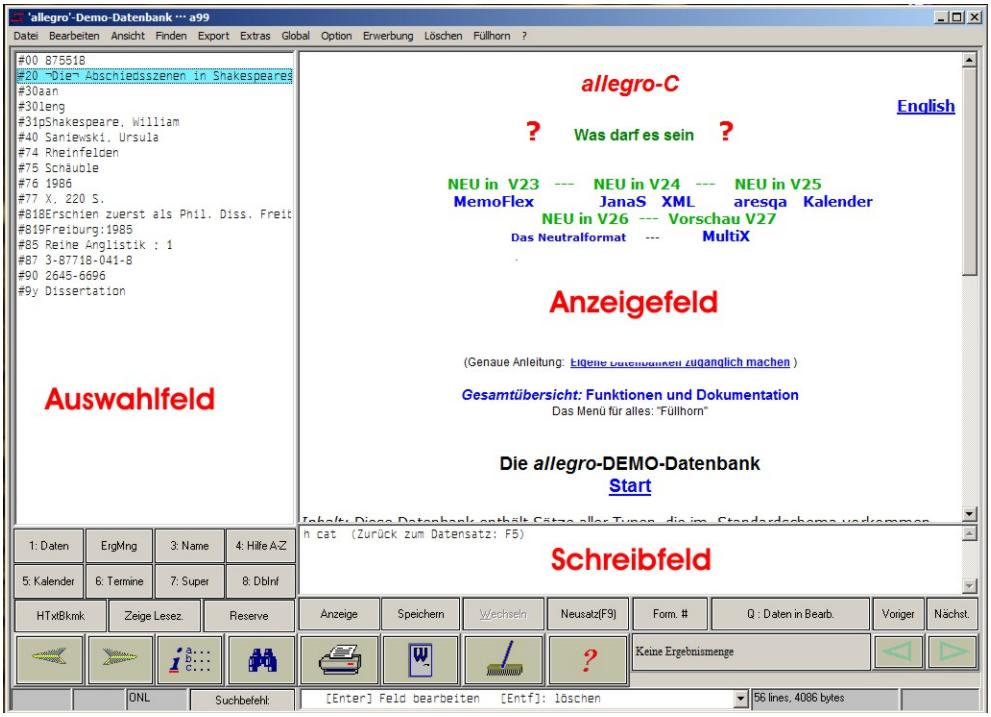

Abbildung 1: Felder von *allegro*

Starten Sie nun Ihr *allegro*-System und geben Sie folgendes in das Schreibfeld ein:

```
x variable "Die Datenbank hat " t " Saetze" \message
```

In dieser kurzen Zeile sind bereits zwei *FLEX*-Befehle enthalten die zum besseren Verständnis fett geschrieben sind. Die *FLEX*-Befehle werden voneinander durch das '****'-Zeichen getrennt, so daß man mit dieser einen Zeile schon zwei Befehle ausführt.

Der erste dieser beiden ist:

```
variable "Die Datenbank hat " t " Saetze"
```

Der Befehl *variable* ist möglicherweise der wichtigste aller *FLEX*-Befehle. Mit diesem Befehl kann man beliebige Zeichenketten miteinander verbinden, anders gesagt, man 'baut' die Zeichenketten zusammen. Die mit *variable* zusammengestellten Zeichenketten befinden sich nach der Ausführung des Befehls in der so genannten *'internen Variablen'* ***iV*** des *allegro*-Systems. Diese interne Variable steht dem System temporär zur Verfügung bis sie durch eine neue Wertzuweisung überschrieben wird, solange kann sie durch andere *FLEX*-Befehle benutzt werden, wie später noch gezeigt werden soll. In diesem Zusammenhang sei auf eine weitere interne Variable hingewiesen: die '*interne Zahlenvariable*' ***iZ.*** Die interne Zahlenvariable enthält Werte, mit denen Rechenoperationen ausgeführt werden können. Doch zurück zum obigen Beispiel, in dem v*ariable* die Zeichenketten "*Die Datenbank hat* " und " *Saetze*" miteinander verbindet. Zu Recht werden Sie sich jetzt fragen, was der Buchstabe ***t*** zwischen den beiden Zeichenketten zu bedeuten hat und warum er nicht auch in Anführungszeichen gesetzt wurde. Die Antwort ist so simpel wie einleuchtend. Eingaben, die zwischen " " oder ' ' eingebettet sind, werden als reiner Text interpretiert und es wird nichts daran verändert, aber außerhalb der Anführungszeichen hat ein Buchstabe eine besondere Bedeutung, die dem Befehl *variable* bekannt ist. Dadurch kann dieser Befehl mehr als nur Texte zusammensetzen! So handelt es sich bei dem Buchstaben *t*, wenn er nicht in Anführungszeichen steht, um eine interne Sondervariable (iS), die die Anzahl der Sätze in der Datenbank angibt. Genau diese Zahl soll uns der Befehl *variable* an der Stelle einsetzen, wo das *t* steht.

Somit haben wir also eine Zeichenkettenfolge aus folgenden einzelnen Bestandteilen erzeugt:

1.	`Die Datenbank hat`	(als Text in "" eingebettet)
2.	`t`	(als numerischen Wert)
3.	`Saetze`	(als Text in "" eingebettet)

Diese wurden durch den eingegebenen Befehl variable miteinander verknüpft, in der internen Variablen abgelegt und warten nun darauf, was mit ihnen geschehen soll.

Es erscheint sinnvoll, daß man sich den Inhalt der von uns erzeugten internen Variablen auf dem Bildschirm anzeigen lässt. Aus diesem Grund steht in der Kommandozeile hinter dem \ der zweite *FLEX*-Befehl: *'message'*

Der `message`-Befehl weist das *allegro* System an, den Inhalt der internen Variablen auf dem Bildschirm in einem kleinen Fenster anzuzeigen. Sie könnten den `message`-Befehl natürlich auch einzeln

durch

```
x message
```

ausführen lassen, doch ist es nicht einfacher die Befehle in einer Zeile zu schreiben?

Ersetzen Sie nun einmal die Befehle *`variable`* und *`message`* durch *`var`* und *`mes`*, so daß im Eingabefeld

```
x var "Die Datenbank hat " t " Saetze" \mes
```

steht.

Man sieht, das Resultat ist identisch! Bis auf eine Ausnahme (dem sehr gefährlichen *`erase`*-Befehl) können alle *FLEX*-Befehle abgekürzt werden, indem man nur die ersten drei Zeichen eingibt, was die Benutzung noch einfacher und schneller macht.

Merke: *FLEX-Befehlen* wird im Eingabefeld ein *x* vorangestellt. Man kann *FLEX*-Befehle bis auf die ersten drei Zeichen abkürzen und diese kombinieren, indem man sie mit dem \ Zeichen miteinander verknüpft.

Kommen wir nun zu einem weiteren wichtigen *FLEX*-Befehl, dem ***`insert`***-Befehl oder kurz ***`ins`***. Geben Sie im Schreibfeld ein:

```
x var "Unwichtiges Buch" \insert #31 \show rec \disp
```

ein.

Den *`var`* Befehl haben wir bereits kennengelernt und daß *FLEX*-Befehle mit dem \ Zeichen verknüpft werden können, wissen wir auch. Doch was machen hier die weiteren *FLEX*-Befehle *`insert`*, *`show`* und *`disp`*?

Betrachten wir den Befehlssatz noch einmal Schritt für Schritt.

1.	`var "Unwichtiges Buch"`	(Weist den Inhalt zwischen "" der *iV* zu)
2.	`insert #31`	(Erzeugt im aktuellen Datensatz die Kategorie #31 und schreibt den Inhalt der *iV* dort hinein)
3.	`show rec`	(Zeigt die aktuellen Datensatzkategorien im Auswahlfeld an)
4.	`disp`	(Zeigt den aktuellen Datensatz auch im Anzeigefeld an)

Sie sehen was passiert ist. Dem aktuellen Datensatz wurde eine Kategorie mit der Nummer 31 hinzugefügt, die den Text "Unwichtiges Buch" enthält. Das Auswahlfeld und das Anzeigefeld zeigen die Änderung an. Das Anzeigefeld wechselt die Hintergrundfarbe von grün auf gelb und unten links erscheint der Text 'EDT', um dem Benutzer anzuzeigen, daß etwas verändert wurde.

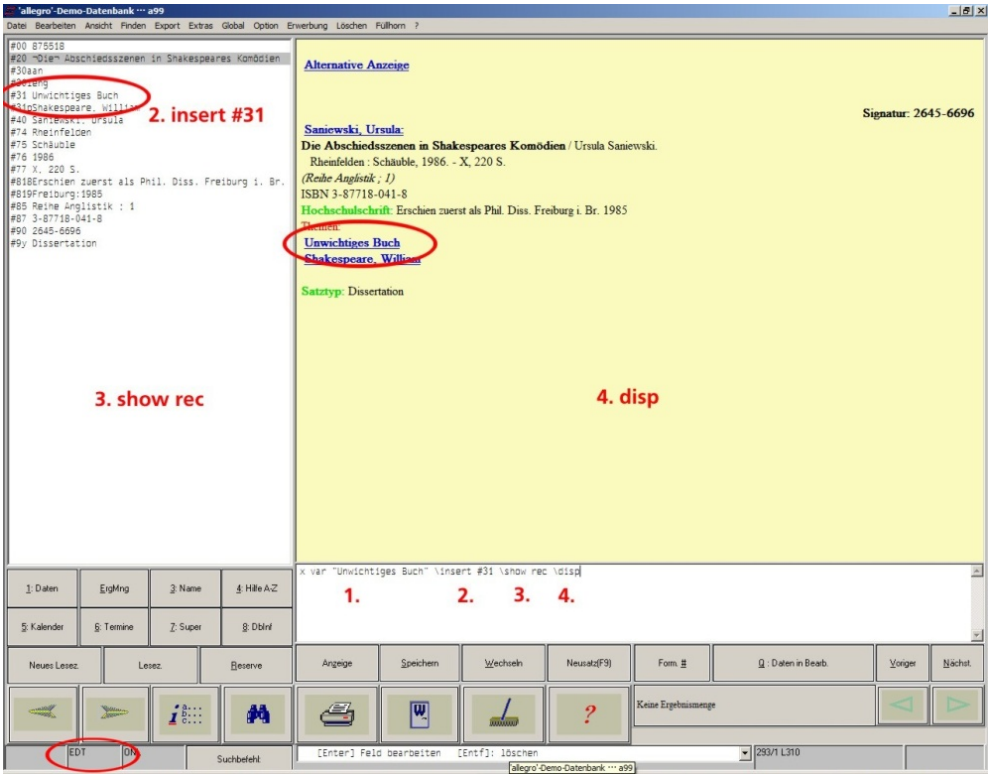

Abbildung 2: Der insert-Befehl

Hinweis: Mit der Tastenkombination 'ALT-w' bzw. der Schaltfläche 'Wechseln' kann man zwischen der geänderten (editierten) und der ursprünglichen Fassung hin- und herschalten!

Zusammen mit dem *var*-Befehl stellt der *insert*-Befehl also ein wichtiges Instrument zur Dateneingabe dar. Mit *var* werden Zeichenketten erzeugt, und mit *ins* weist man diese Zeichenketten Kategorien zu. Der *ins*-Befehl kann aber noch mehr!

Mit dem *ins*-Befehl kann man nicht nur den Inhalt der *iV* in eine Kategorie schreiben, man kann mit dem *ins*-Befehl auch den Inhalt der *iV* in dauerhafte Variablen kopieren. Diese Variablen beginnen mit ***#u*** und zwei beliebigen darauf folgenden Zeichen. Dabei ist zu beachten, daß der Name der Variablen 'case sensitive' ist, das bedeutet, daß zwischen *GROSS* und *klein*-Schreibung unterschieden wird. So sind beispielsweise ***#u****AB*, ***#u****Ab*, ***#u****aB* und ***#u****ab* vier unterschiedliche Variablen!

Alternativ können Sie bei der Eingabe von *FLEX*-Befehlen statt des vorangestellten '*x*' auch eine Zahl ***i*** (***i=0***...***9)*** eingeben, dann wird der Befehl nicht sofort ausgelöst, sondern stattdessen in der Benutzervariablen ***#uXi*** gespeichert. Der Inhalt der Benutzervariablen *#uXi* wird durch die Eingabe der Tasten *ALT+i* ausgelöst! Beachten Sie bitte dabei, daß standardmäßig die *FLIP*-Buttons (siehe Abbildung 3) mit den Benutzervariablen *#uX1* bis *#uX8* belegt sind, und diese gegebenenfalls überschrieben werden könnten!

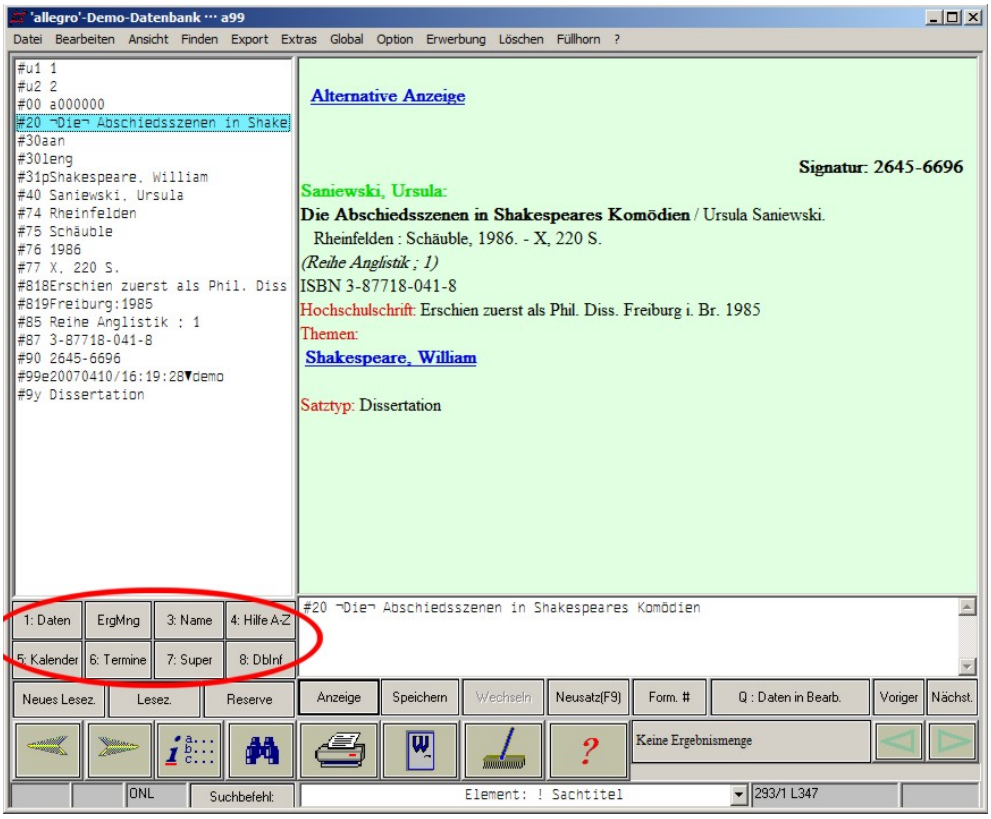

Abbildung 3: Die FLIP-Buttons

Somit ist dem Anwender eine gute und schnelle Möglichkeit gegeben, wiederkehrende Befehlssequenzen in *FLEX* auf Tastaturkürzel zu legen oder einem der 8 *FLIP*-Buttons zuzuweisen.

An dieser Stelle eine kurze Anmerkung zu den *FLIP*-Buttons oder *FLIP*s allgemein:

Man kann *FLIP*s als eine Art Hyperlink beschreiben, mit dem man andere Texte, Befehle oder Programme aufruft. Diese Programme sind dann ihrerseits wieder *FLEX*e. Für den "normalen" Endanwender soll es an dieser Stelle bei dieser kurzen Erklärung bleiben. Einen ausführlicheren Text kann man mit `'h flip'` abrufen.

3 *FLEX*-Befehle und interne Sondervariablen

Die beiden vorangegangenen Beispiele haben gezeigt, wie generell mit den *FLEX*-Befehlen im Eingabefeld umzugehen ist. Sie haben die internen Variablen *iV* und *iZ* kennengelernt und wissen wie man *FLEX*-Befehle innerhalb einer Zeile kombiniert. Doch noch immer stellt sich die Frage welche *FLEX*-Befehle und welche internen Sondervariablen, wie das bereits kenengelernte *t*, gibt es im *allegro* System und wie kann man sich die Gesamtheit(en) davon anzeigen lassen?

3.1 Anzeige der FLEX-Befehle

Um sich die Liste der verfügbaren *FLEX*-Befehle inklusive der assoziierten Erläuterungen anzeigen zu lassen, gibt es für den Anwender verschiedene Möglichkeiten. Diese sind dem Kenntnisstand des jeweiligen Anwenders entsprechend hilfreich einsetzbar. Folgende Vorgehensweisen sind zum Aufrufen der Dokumentation der *FLEX*-Befehle von allgemein bis hin zu speziell empfehlenswert:

3.1.1 Einfacher, allgemeiner Aufruf

Für den Anfänger empfiehlt es sich, die Befehlsübersicht mit dem Befehl ***view*** aufzurufen. Dazu gibt man im Schreibfeld den Befehl

```
v flex
```

ein, wobei das *v* für *view* steht. Das Resultat der Eingabe ist das in Abbildung 4 dargestellte Fenster.

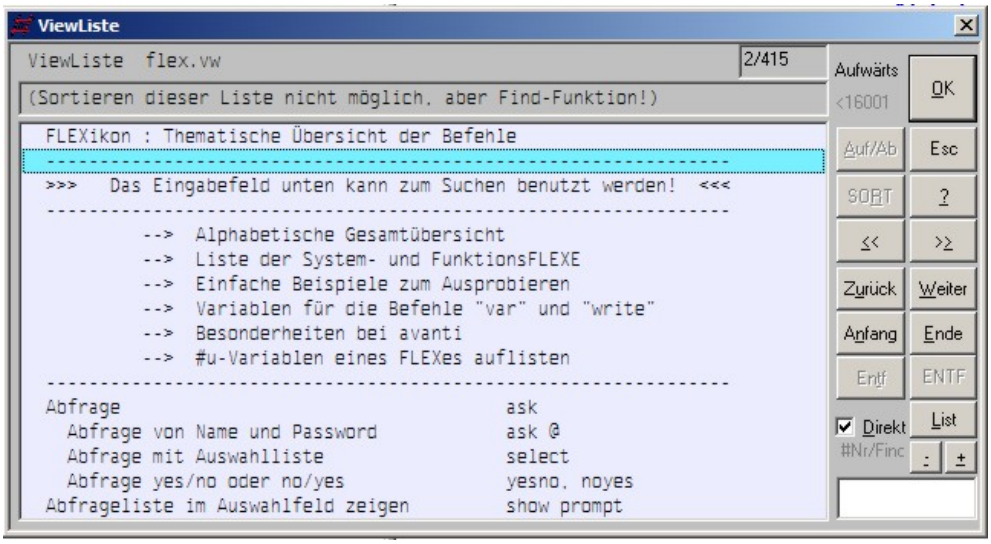

Abbildung 4: Befehlsliste mit v flex

Hier bekommt der Anwender eine thematische Übersicht, der zur Verfügung stehenden Befehle. Sogar eine Suchfunktion ist integriert, so daß das Auffinden eines benötigten *FLEX*-Befehls erleichtert wird. Die Dokumentation des jeweiligen Befehls wird im Anzeigefeld von *allegro* erscheinen, sowie der Balken die Zeile auswählt.

3.1.2 Aufruf des alphabetischen Registers

Ist der Anwender schon weiter fortgeschritten, so wird er häufiger auf das knappe alphabetische Register zurückgreifen wollen. Man erhält es mittels Eingabe des Befehls

h *xa*

im Schreibfeld. Der Buchstabe *h* teilt dem *allegro* System mit, daß die Hilfedatei für den darauf folgenden *FLEX*-Befehl aufgerufen werden soll. In diesem Fall wird der Teil des *FLEX*ikons (siehe Kapitel 8) aufgerufen, der das alphabetische Register aller *FLEX*-Befehle enthält. Innerhalb dieses Registers kann mit einem Klick auf den jeweiligen Befehl die entsprechende Dokumentation aufgerufen werden.

3.1.3 Direkte Abfrage der Befehlsdokumentation

Noch etwas spezieller ist schließlich die direkte Abfrage der Befehlsdokumentation. Hier fragt man für einen konkreten Befehl die Hilfedatei über eine Eingabe im Schreibfeld direkt ab. In solchen Fällen findet folgender Befehl Anwendung

h *x<FLEX-Befehlsname>.*

Man umgeht damit den 'Umweg' über das komplette Register den entsprechenden Befehl zu suchen und ruft stattdessen für den gewünschten Befehl die Dokumentation auf. Wichtig hierbei ist, daß zwischen dem *x* und dem *FLEX*-Befehl KEIN Leerzeichen steht!

3.2 Anzeige der internen Sondervariablen

Mit Hilfe der internen Sondervariablen (iS) kann der aktuelle Status des Systems abgefragt werden. Dies können bestimmte Eigenschaften oder auch Einstellungen sein. Um die Liste der internen Sondervariablen zu erhalten, verfährt man ebenso wie bei der Abfrage der Liste der *FLEX*-Befehle. Die Eingabe des Befehls

h *xcstring*

im Schreibfeld ruft im Anzeigefeld die Übersicht über die "*code string*"-Syntax auf. Darin enthalten (unter Punkt 5) sind die Informationen über die internen Sondervariablen (siehe Abbildung 5).

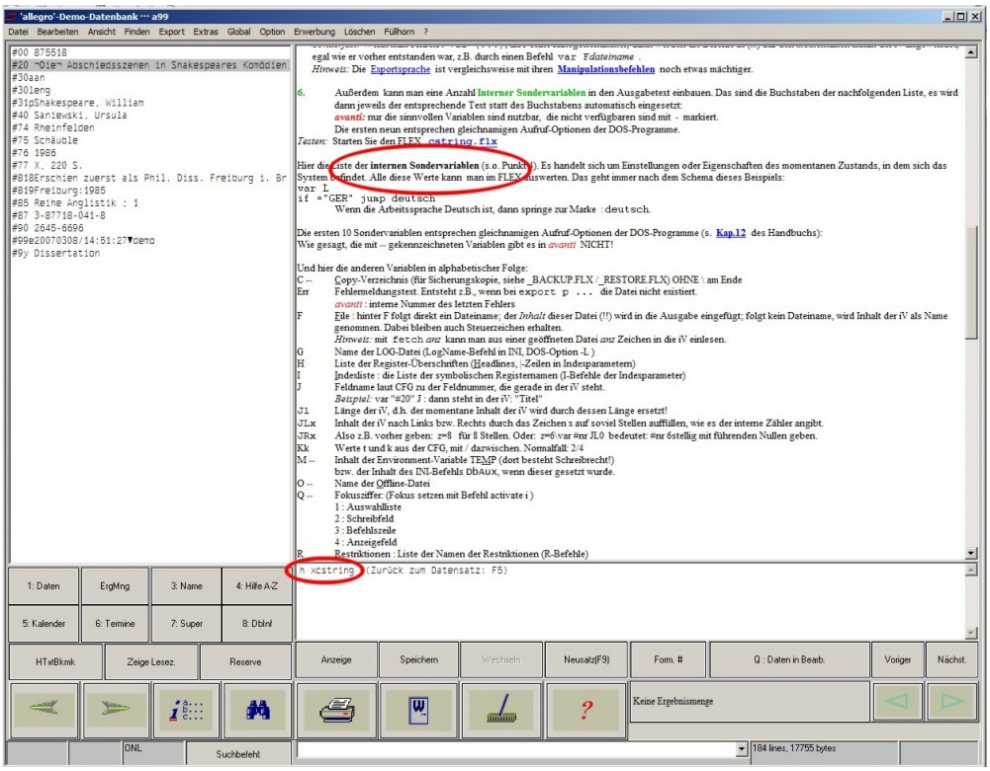

Abbildung 5: Aufruf der Liste der internen Sondervariablen

Innerhalb dieser Liste sind sämtliche internen Sondervariablen alphabetisch geordnet von ***C*** (wie: Copy Verzeichnis) bis ***zv*** (wie: Nummer der oben ersten sichtbaren (visible) Zeile).

Die Auswertung der internen Sondervariablen innerhalb eines *FLEXes* erfolgt immer nach folgendem Schema:

```
var <interne Sondervariable>
if =<"Bedingung"> jump <Sprungziel>
```

Ein konkretes Beispiel hierzu ist die Abfrage der momentan eingestellten Sprache. Der Liste der internen Sondervariablen (siehe Kapitel 9.3) entnimmt man, daß der Buchstabe 'L' für die Sprachenbezeichnung steht. Somit ergibt sich für die Abfrage der momentanen Spracheneinstellung folgende Syntax:

```
var L
if ="GER" jump deutsch
```

Ist die voreingestellte Sprache also deutsch (GER) so werden in der weiteren Befehlsabfolge des *FLEXes* die Befehle, die nach der Sprungmarke ':deutsch' stehen abgearbeitet. Wie diese Sprungmarke definiert ist, folgt in einem späteren Kapitel.

Man kann die internen Sondervariablen auch im Schreibfeld benutzen. Um das vorangegangene Beispiel aufzugreifen, wird die eingestellte Sprache durch Eingabe von

```
x var L \mes
```

im Schreibfeld abgefragt und dargestellt. Im Normalfall werden Sie die internen Sondervariablen aber innerhalb eines *FLEXes* verwenden!

Merke: Mit den internen Sondervariablen kann der Systemstatus abgefragt werden. Dies können sowohl Systemeinstellungen wie auch Systemeigenschaften sein!

3.2.1 Der Code String *cstring*

An dieser Stelle erscheint aufgrund seiner Relevanz im *FLEX*-Konzept ein kleiner Exkurs zum Thema cstring notwendig. Erst im Zusammenhang mit dem cstring erreichen die Befehle *var* und *wri* ihre wahre Mächtigkeit. Ein *cstring* folgt jeweils diesen *FLEX*-Befehlen und kann sich aus verschiedenen, aneinander geketteten Code- Elementen in beliebiger Reihenfolge zusammensetzen. Diese Code-Elemente bestehen ihrerseits aus sechs unterschiedlichen Typen, namentlich:

1. Zeichenfolgen: Dies sind alle in **" "**, **' '** oder **| |** eingeschlossene, druckbare Zeichen. (Jedes **** muß darin verdop pelt werden)
2. ASCII-Codes: Diese werden als Dezimalzahlen angegeben. Zum Beispiel 13 10 für einen Zeilenvorschub.
3. Kategorieninhalte: Vom Typus **#nnn**, wobei nnn auch eine Son derkategorie oder eine #u Kategorie sein kann.
4. Freie Variablen: Vom Typus **$name**.
5. int. Sondervariablen: Einzelne Buchstaben, siehe Tabelle 3 in Kapitel 9.3
6. Manipulationsbefehle: Befehle in **( )** gesetzt

Beispiele für die unter Punkt 6 genannten Manipulationsbefehle finden Sie, wenn Sie im Schreibfeld von *allegro* den Befehl

```
h mbtest
```

eingeben und sofort interaktiv ausprobieren.

4 Benutzung von fertigen *FLEX*en

FLEX ist jedoch weitaus mehr als nur die Eingabe von Befehlen oder verknüpften Befehlsketten im Eingabefeld von *allegro*. Wie bereits einleitend beschrieben, besteht ein *FLEX* aus einer Vielzahl von *FLEX*-Befehlen, die abgearbeitet werden. Die bislang beschriebene Vorgehensweise, im Schreibfeld einzelne oder verkettete *FLEX*-Befehle einzugeben, eignet sich gut als Anschauungsbeispiel zum Testen und Ausprobieren.

Normalerweise werden die *FLEX*-Befehle aber zeilenweise in eine Datei geschrieben, die dann die Endung flx erhält. *FLEX*e sind also reine Textdateien, die mit einem Editor ihrer Wahl bearbeitet werden können (mehr dazu siehe Kapitel 6).

Wie Sie bereits erfahren haben, liefert *allegro* einige fertige *FLEX*e mit, die beispielsweise aus ViewListen oder aus Hilfetexten automatisch aktiviert werden. So wird zum Beispiel beim Betätigen des 'Drucken'-Buttons der *FLEX onprint.flx* ausgelöst. Alle diese mitgelieferten und vorgefertigten *FLEX*e finden Sie im Unterverzeichnis '\flex' ihrer *allegro*-Installation. Jede dieser Dateien ist mit Kommentaren versehen, damit Sie den Ablauf des *FLEX*es besser nachvollziehen können.

Man kann einige *FLEX*e aber auch von Hand in Betrieb setzen. Welche *FLEX*e dies genau sind, ist der Tabelle 4 in Kapitel 9.4 zu entnehmen. Um einen *FLEX* von Hand zu starten, gibt man im Schreibfeld als Präfix ein '***X***' ein und darauf folgend den Namen des *FLEX*es, der ausgeführt werden soll. Hierbei ist es wichtig, daß es ein großes '*X*' ist, da die folgende Zeichenkette ansonsten als simpler *FLEX*-Befehl und nicht als *FLEX* interpretiert werden würde.

Geben Sie nun im Schreibfeld einmal folgenden Befehl ein:

```
X kalender
```

Das *allegro* System erkennt am groß geschriebenen '*X*', daß es sich hierbei um einen *FLEX* handelt, der ausgeführt werden soll und sucht in folgenden Verzeichnissen nach dem angegebenen *FLEX*:

1. Datenbankverzeichnis: <DbDir>
2. Unterverzeichnis \flex im Programmverzeichnis: <ProgDir>\flex
3. Programmverzeichnis: <ProgDir>
4. Startverzeichnis

Dort, wo *allegro* die Datei zuerst findet, wird sie verwendet.

Merke: *FLEX*e sind reine Textdateien, in denen zeilenweise *FLEX*-Befehle abgearbeitet werden. Einige der mitgelieferten *FLEX*e können auch manuell gestartet werden. Dies geschieht, indem man dem *FLEX* ein 'X' und ein Leerzeichen im Schreibfeld voranstellt. Das *allegro*-System hat fest definierte Verzeichnisse, in denen es den aufgerufenen *FLEX* sucht!

Hat das System den *FLEX* `kalender.flx` gefunden, so wird er ausgeführt und Sie sollten im Anzeigefeld von *allegro* folgendes dargestellt bekommen.

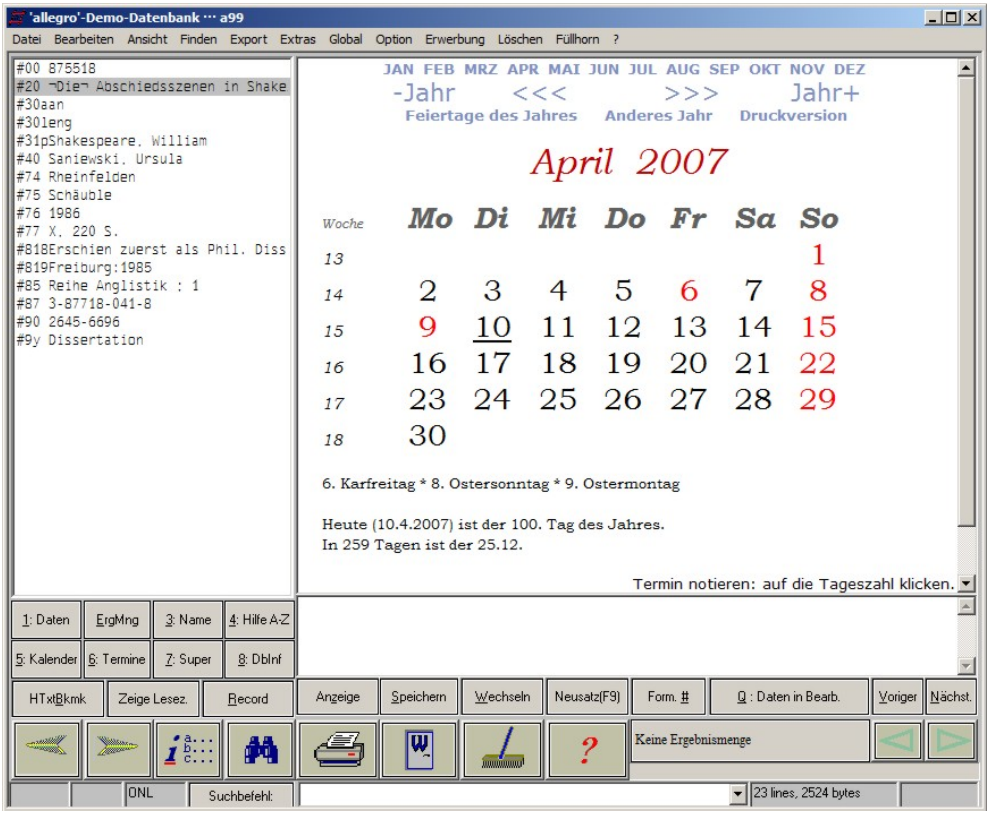

Abbildung 6: Der *FLEX* kalender.flx

Sie sehen, wie vielfältig *FLEX* einsetzbar ist. Mit diesem *FLEX* ist eine komplette Kalenderfunktion im *allegro* System integriert!

Der Quelltext des *FLEX*es `kalender.flx` ist jedoch für den Anfänger noch etwas zu komplex, als daß er an dieser Stelle ausführlich kommentiert werden könnte!

5 Die Syntax von *FLEX* anhand von einfachen Beispielen

Wie im vorangegangenen Kapitel bereits erwähnt, besteht ein *FLEX* aus einer Abfolge von *FLEX*-Befehlen und - im Idealfall - Kommentarzeilen in einer Textdatei. Dieses Kapitel soll Ihnen, anhand von einfachen ausgewählten Beispielen, die Syntax von *FLEX* näher bringen.

5.1 Beispiel 1: Der FLEX `_new.flx`

Dieser *FLEX* löst die Erfassung eines Datensatzes aus (vergleiche dazu Tabelle 4 in Kapitel 9.4). *`_new.flx`* ist ein *FLEX*, der nicht manuell ausgelöst werden kann. Der dazugehörige Quelltext ist relativ kurz:

```
  _NEW.FLX : Neuanlage einer Datenbank (Start automatisch aus a99)
  2000-06-12
  Vorhanden sein muss ein Verzeichnis fuer die Daten, eine INI-Datei (in der
  das Verzeichnis eingetragen ist), eine CFG und alle noetigen Parameter

  Wenn es alcarta ist, nichts machen!
variable m
if ="alc" end

#00 a000000
#u1 1
#u2 2
put
```

Der eigentliche Programmcode, sprich die *FLEX*-Befehle und die entsprechenden Attribute, ist in diesem Beispiel zur besseren Erklärung **blau** markiert während die Kommentarzeilen **grau** angezeigt werden. Man kann erkennen, daß die Kommentarzeilen eingerückt sind, da sie mit einem Leerzeichen beginnen. Ein Leerzeichen zu Zeilenbeginn oder auch zwei aufeinander folgende Leerzeichen innerhalb einer Zeile werden von allegro automatisch als Kommentarzeichen interpretiert, alternativ können Sie auch '//' als einleitende Zeichen einer Kommentarzeile schreiben. Welche von diesen Möglichkeiten Sie bevorzugen, bleibt Ihnen überlassen, an dieser Stelle möchten wir Ihnen jedoch, weil es besser lesbar ist, die Variante mit den beiden '***//***' Zeichen empfehlen. Betrachten wir nun den *FLEX*, er beginnt, mit einem einleitenden Kommentar:

```
  _NEW.FLX : Neuanlage einer Datenbank (Start automatisch aus a99)
  2000-06-12
  Vorhanden sein muss ein Verzeichnis fuer die Daten, eine INI-Datei (in der
  das Verzeichnis eingetragen ist), eine CFG und alle noetigen Parameter

  Wenn es alcarta ist, nichts machen!
```

Nachdem im Kommentar beschrieben wurde was folgt, kommt der erste Befehl:

`variable m`.

Es soll geprüft werden, um welche Programmversion es sich handelt. Dazu benötigt man die interne Sondervariable '***m***' (siehe Tabelle 3, Kapitel 9.3), deren '*Wert*' in die *iV* geschrieben wird. Entspricht dieser Wert der Zeichenfolge '***alc***', also ist es das Programm alcarta, dann soll nichts gemacht werden und der *FLEX* wird beendet (***end***). Die Abfrage der *iS* und die Überprüfung des Wertes werden in den ersten beiden Befehlszeilen geleistet:

```
variable m
if ="alc" end
```

Entspricht der Wert nicht '*alc*', so weist der *FLEX* den Kategorien

#00 (*Identifikationsnummer*)
#u1 (*Sonderkategorie "Überschrift 1"*)
und
#u2 (*Sonderkategorie "Überschrift 2"*)

Werte zu und speichert diese für den ersten Satz (***put***). Danach wird der *FLEX* beendet.

```
#00 a000000
#u1 1
#u2 2
put
```

Es ist möglich, daß in Ihrem Quelltext noch weitere Befehlszeilen nach der '*put*' Anweisung folgen, diese stammen aus einer alten Version und sind nicht mehr wichtig. Deshalb steht die Anweisung '*end*' in der folgenden Zeile. Das '*end*' ist sozusagen die 'Ausstiegsmarke' des *FLEX*es. Wird diese Stelle erreicht, wird der *FLEX* beendet, egal was unter dieser Zeile noch kommt. Seien Sie also nicht zu sehr verwundert, wenn Sie möglicherweise noch weitere Anweisungen nach dem '*put*'-Befehl finden. Durch die '*end*'-Anweisung werden diese Zeilen nicht ausgeführt!

5.2 Beispiel 2: Der FLEX notepad.flx

Dieser *FLEX* ist im Vergleich zum vorhergehenden schon ein wenig komplexer. Wenn Sie möchten, können Sie ihn testen, er ist manuell startbar. Wie ein *FLEX* manuell gestartet wird, haben Sie in Kapitel 4 erfahren. Auch hier finden Sie die Kommentarzeilen **grau** und die Anweisungszeilen **blau** markiert. Die Kommentarzeilen beginnen hier mit den im ersten Beispiel beschriebenen // Zeichen.

Der gesamte *FLEX* besteht aus insgesamt 5 Schritten, die nach und nach abgearbeitet werden. Im ersten Schritt werden die Exportparameter '*e*' in die *iV* geschrieben und danach mit '*ins #uwe*' der Variablen *#uwe* zugewiesen.

```
// NOTEPAD.FLX : Externes Editieren des Satzes mit Notepad
// 2001-02-09

// Exportparam sichern
variable e
insert #uwe
```

Im zweiten Schritt wird der entsprechende Satz für die externe Bearbeitung vorbereitet. Dazu wird die zunächst die Exportparameterdatei e-w0 gesetzt. Im folgenden Schritt wird die Tabelle asciansi hinzu geladen, um die Zeichenumwandlung von ASCII nach ANSI zu aktivieren. Die dritte Zeile öffnet die zu bearbeitende Datei notep.txt mit dem X-Editor. Der aktuelle Satz wird nun exportiert und die mit dem X-Editor geöffnete Datei wieder geschlossen.

```
// Satz mit e-w0 ausgeben in notep.txt
export[1] param e-w0
export[1] table asciansi
open x notep.txt
export[2]
close x
```

[1],[2]: Die Befehle `xport` und `download` sind ab Version 27.1 zusammengefasst zu `export`. Die Funktionalität der alten Befehle bleibt erhalten!

Nachdem nun im Hintergrund die Datei notep.txt erstellt und mit den entsprechenden Daten des Datensatzes gefüllt wurde, wird sie im folgenden Schritt mit einem externen Programm (hier notepad) aufgerufen. Dazu dient der '*call*'-Befehl.

```
// Notepad aufrufen (evtl. hier anderen Editor einsetzen)
call notepad notep.txt
```

Hat man schließlich die entsprechenden Änderungen vorgenommen und mit Notepad oder einem Editor eigener Wahl gespeichert, müssen diese Veränderungen des Datensatzes dem System zugeführt werden. Dies geschieht im folgenden Schritt. Die veränderte Datei notep.txt wird vom System und die Hilfsdatei *e.adt*, die als Schnittstelle zur Übermittlung des Datensatzes an das System dient, mit dem X-Editor geöffnet. Der eingelesene Inhalt des veränderten Datensatzes wird mittels des '*fetch*'-Befehls und einem darauf folgenden Zahlenwert, der eine gewünschte Größe in Byte angibt, in die *iV* geschrieben, bzw. an eventuell vorhandene Daten in der *iV* angehängt. Als nächstes erfolgt die Wandlung des Satzes vom ANSI in das ASCII Format und der Inhalt der *iV* wird in die Ausgabedatei geschrieben. Dem Schließen des X-Editors folgt das Neueinlesen des Datensatzes und die temporäre Datei notep.txt wird wieder gelöscht.

```
// Satz wieder einlesen (ANSI!)
open notep.txt
open x e.adt
fetch 6000
asci
write ^
close x
read
delete notep.txt
```

Abschließend werden die Exportparameter, die in Schritt 1 gesichert wurden, zur Wiederherstellung aus der Benutzervariablen *#uwe* in die *iV* zurückgeschrieben. Zum Abschluß werden die Parameterdatei, in der Ansicht ein Lesezeichen auf den aktuellen Datensatz und der Anzeigemodus des Anzeigefeldes gesetzt.

```
// Exportparam wiederherstellen
variable #uwe
export param
set b
set d
```

5.3 Beispiel 3: Der FLEX `_start.flx`

Dieser *FLEX* wird gleich nach dem Start von *allegro* ausgeführt. Er wirkt ein wenig komplexer als die vorhergehenden, Sie werden jedoch schnell bemerken, daß Sie auch vor solchen *FLEX*en nicht zurückschrecken brauchen. Dieser *FLEX* dient zum Initialisieren der Speicherbereiche, zur Belegung der FLIP-Buttons und zur Überprüfung des Vorhandenseins der notwendigen Schriftenarten. Gehen wir diesen Ablauf ebenso Schritt für Schritt durch.

Zunächst wird der erste Datensatz geladen um die erforderlichen Speicherbereiche zu initialisieren. Ausgelöst wird dies durch die '`find`' Anweisung mit der darauf folgenden Angabe der Satznummer. Die '`show rec`' Anweisung initiiert die Anzeige des Satzes in der Auswahlliste (sonst erscheint er nur im Anzeigefeld).

```
// START.FLX : wird direkt nach Start ausgefuehrt
// 2001-02-07
// Satz 1 laden und zeigen
// (nur um die Speicherbereiche zu initialisieren)

Find #1
show rec
```

Damit ist der erste Teil, die Initialisierung des Speicherbereiches abgeschlossen. Im darauf folgenden Codesegment werden die *FLIP*-Buttons 1 bis 8 mit Funktionen belegt. Die Syntax zur Belegung der Buttons läßt sich wie folgt erläutern. Es beginnt mit der Anweisung '`flip`', der Nummer des *FLIP*-Buttons und darauf folgenden Attributen. Diese Attribute umfassen den Button-Text (eingebettet in & und =) und eine darauf folgende Anweisung, die ausgeführt werden soll, wenn der *FLIP*-Button gedrückt wird. Der *FLIP*-

Button Nummer 5 erhält beispielsweise den Text '*5: Kalender*' und löst beim Anklicken den Befehl '`X kalender`', also den Aufruf der Kalenderfunktion, aus.

```
// Belegung der Flip-Tasten
flip 1&1: Daten=x sho rec
flip 2&2: ErgMng=x sho erg
flip 3&3: ErgName=X rset
flip 4&4: QUICK=x View quick
flip 5&5: Kalender=X kalender
flip 6&6: Schema=x sho cfg
flip 7&7: Super=h super
flip 8&8: DbInfo=X _dbinfo
```

Den nächsten Teil sollten Sie wiedererkennen, Sie haben einen ähnlichen Abschnitt bereits im ersten Beispiel kennengelernt. Es werden der aktuelle Programmname und die aktuelle Programmversion abgefragt. Ist das aktuelle Programm alcarta, so soll die Variable *#uaV1* gesetzt werden, ansonsten passiert an dieser Stelle nichts.

```
// Wenn alcarta, dann Variable #uaV setzen (d-wrtf.apr)
variable m
if "alc" #uaV1
```

Im nächsten größeren Abschnitt soll geprüft werden, ob die für das System notwendigen Schriftarten installiert sind. Dazu wird zunächst der Pfad des Windows-Verzeichnisses abgefragt und dessen "Wert" in die *iV* kopiert Diesem Wert wird durch die folgende *var* Anweisung der Pfad zur Schriftartendatei a-letter.ttf hinzugefügt und das Datum der letzten Aktualisierung abgefragt. Ist die Schriftart vorhanden (dann gilt die Bedingung *'not no'*) so soll der *FLEX* zur Sprungmarke *`:fertig`* springen. Existiert diese Schriftart nicht (Bedingung *'no'* gilt dann), wird der folgende Text in einem Ausgabefenster angezeigt:

> *Auf Ihrem PC sind die allegro-Schriften nicht installiert*
> *Empfehlung: Damit Index und Kurzliste brauchbar aussehen,*
> *stellen Sie über 'Optionen / Datenfont' die Schrift 'Courier New' ein*
> *Oder lassen Sie sich die allegro-Schriften installieren*
> *(Die TTF-Dateien liegen auf C:\allegro)*

```
// Pruefen, ob Schriften vorhanden
get env windir
variable +"\\fonts\\a-letter.ttf"
ftime
if not no jump fertig
variable "Auf Ihrem PC sind die allegro-Schriften nicht installiert"
variable + n n "Empfehlung: Damit Index und Kurzliste brauchbar aussehen,"
variable + n "stellen Sie ber 'Optionen / Datenfont' die Schrift 'Courier New' ein"
variable + n n "Oder lassen Sie sich die allegro-Schriften installieren"
variable + n "(Die TTF-Dateien liegen auf " P ")"
ansi
message
:fertig
```

Abschließend können noch einige Optionen für die automatische Aktivierung einzelner Module beim Programmstart vorgenommen werden. Dies geschieht, indem man die betreffenden Zeilen ganz nach links rückt.

```
// Jetzt noch einige Einstellungen, die man wahlweise aktivieren kann.
// Aktivieren heisst: eine Zeile ganz nach links setzen
// (denn sonst wird sie nicht ausgefuehrt!)
// naechste Zeile aktivieren, wenn ALFA sofort aktiviert sein soll:
 #uAL1
// naechste Zeile aktivieren, wenn ALFA-Ausleiher angezeigt werden soll
 #uAU2
// bzw. diese zwei Zeilen, wenn er in alcarta nicht gezeigt werden soll
 variable m
 if not "alc" #uAU2
// Aktivieren Sie diesen Befehl, wenn beim Start breite Anzeige aktiv sein soll
 set db
```

6 Eigene *FLEX*e erstellen

Nachdem Sie nun einige Theorie über sich haben "ergehen" lassen müssen, ist es an der Zeit, zu dem Punkt zu kommen an dem Sie Ihren ersten eigenen *FLEX* erstellen und in Ihr System integrieren. Zum Erstellen einer *FLEX*-Datei benötigen Sie einen Editor Ihrer Wahl oder Sie erstellen die Datei direkt aus *allegro* heraus. Empfehlenswert und hilfreich ist die Benutzung eines Editors mit der Möglichkeit des sogenannten Syntax-Highlighting, wie es z.B. bei dem Open Source Editor Notepad++ der Fall ist. Die NPP-Erweiterung des Syntax-Highlightings für *FLEX* ist bereits von uns erstellt worden und kann beliebig erweitert und verändert werden. Möchten Sie den *FLEX* mit Hilfe von *allegro* erstellen, so rufen Sie das Füllhorn (ALT+h) auf, und wählen den Punkt "Neue Textdatei schreiben" an (siehe Abbildung 7).

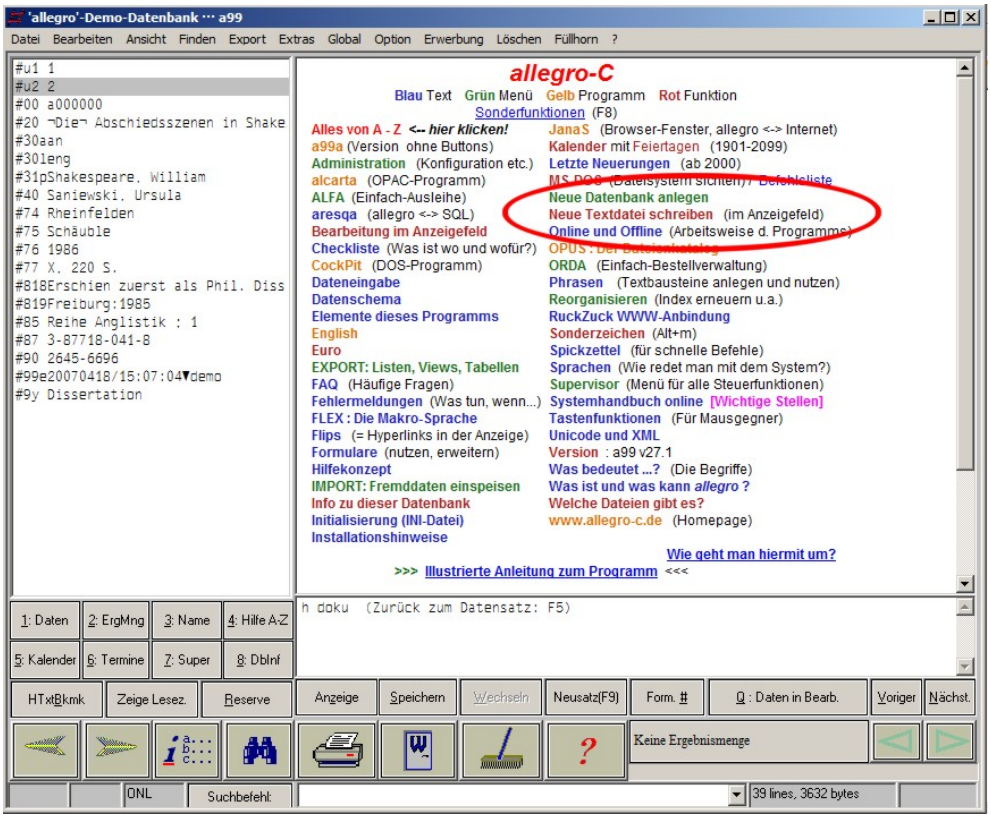

Abbildung 7: Füllhorn - Neue Textdatei schreiben

Erstellen wir doch einmal anhand eines konkreten Beispiels gemeinsam eine *FLEX*-Datei. Als Beispiel soll unser neuer *FLEX* die aktuelle Ergebnismenge durcharbeiten und dabei in jedem Datensatz ein Feld mit der Nummer #31 anlegen, daß ein einzugebendes Schlagwort enthält. Der entsprechende *FLEX* hätte dann folgenden Inhalt:

```
// SW-NEUB.FLX : Neues Schlagwort in jeden Satz der Erg.Menge einfuegen
// 2005-03-14  Variante B: Anderes SW fuer jeden Satz
// Gibt es ueberhaupt eine Erg.Menge? Wenn nein -> :keine
if empty jump keine

// Ersten Satz der Erg.Menge laden
first

// Folgende Schleife wird fuer jeden Satz ausgefuehrt:
:schleife

// Den Satz anzeigen
show rec
display

// Das gewuenschte neue Schlagwort abfragen:
// (Wenn #uSW schon belegt, erscheint es als Vorgabe)
ask Wie soll das Schlagwort fuer diesen Satz lauten?=#uSW
if "" end

// und in Hilfsvariable #uSW einfuegen
insert #uSW

// Schlagwort in #31 einsetzen, aber wenn #31 schon besetzt, dann
// das neue hinten mit "; " anhaengen
if #31 var #31 "; " #uSW
if not #31 var #uSW
insert #31

// und Satz wieder speichern:
put

// naechsten Satz laden:
next

// Wenn es noch einen gab, dann Sprung nach :schleife
if yes jump schleife

// Sonst ist Schluss:
jump schluss
:keine

// Das gewuenschte neue Schlagwort abfragen:
ask Wie soll das Schlagwort fuer diesen Satz lauten?=#uSW
if "" end

// und in #uSW speichern
ins #uSW

// dann an #31 anhaengen
if #31 var #31 "; " #uSW
if not #31 var #uSW
insert #31

// und Satz wieder speichern
put

:schluss

// Den momentanen (also letzten) Datensatz noch neu anzeigen
show rec
display
message Erledigt!
```

Sie können diesen Text direkt kopieren (Markieren und STRG+C) und in den entsprechenden Editor einfügen (STRG+V), um sich die Schreibarbeit zu ersparen. Ja nachdem für welche Eingabevariante Sie sich entschieden haben, sollten Sie entweder Abbildung 8 oder ein mit Abbildung 9 vergleichbares Bild auf Ihrem Bildschirm sehen.

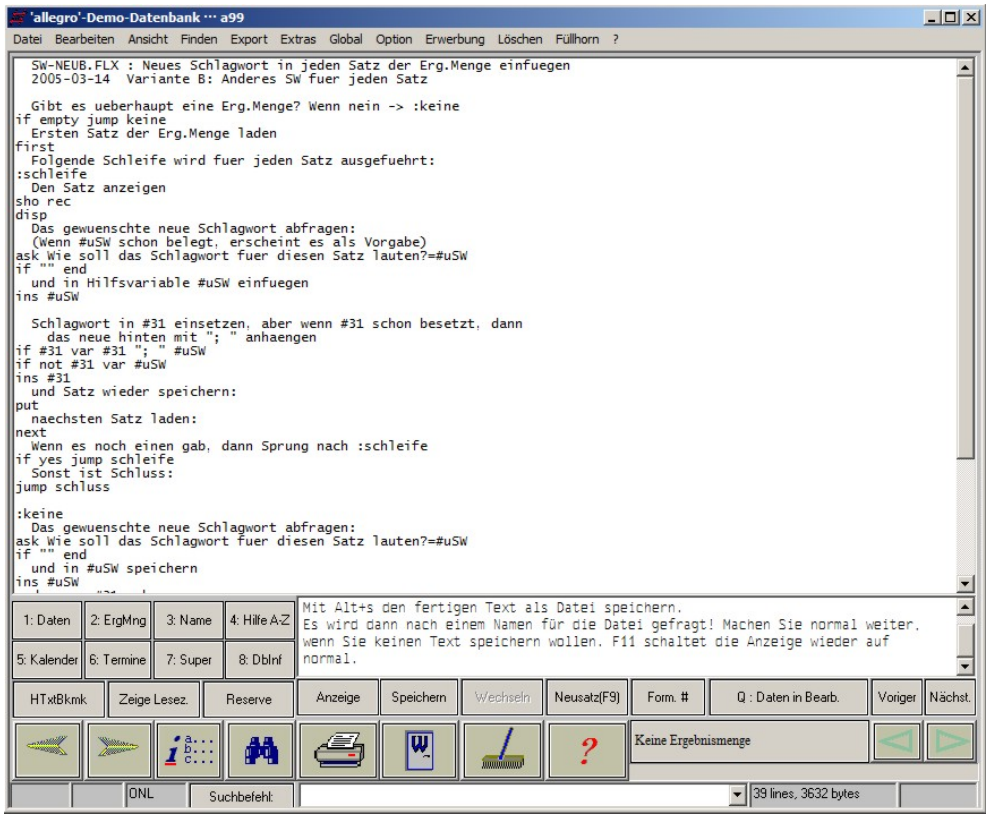

Abbildung 8: sw_neu.flx mit *allegro*

In Abbildung 9 ist der Vorteil des Syntax-Highlightings erkennbar. Die *FLEX*-Befehle, die Variablen, die Kommentarzeilen und der normale Text sind stilistisch voneinander gut zu unterscheiden, so daß man sich besser und schneller im *FLEX* zurechtfinden kann. Dies ist besonders bei größeren *FLEX*en sehr hilfreich!

Nachdem nun der Quellcode im Editor eingegeben wurde, muß die Datei lediglich nur noch in einem Pfad wie in Kapitel 4 beschrieben, abgespeichert werden, zum Beispiel unter dem Namen `sw_neu-flx`.

Zum Testen können Sie nun im Schreibfeld die Anweisung

```
X sw_neu
```

eingeben, um Ihren ersten, selbst geschriebenen *FLEX* einmal zu testen.

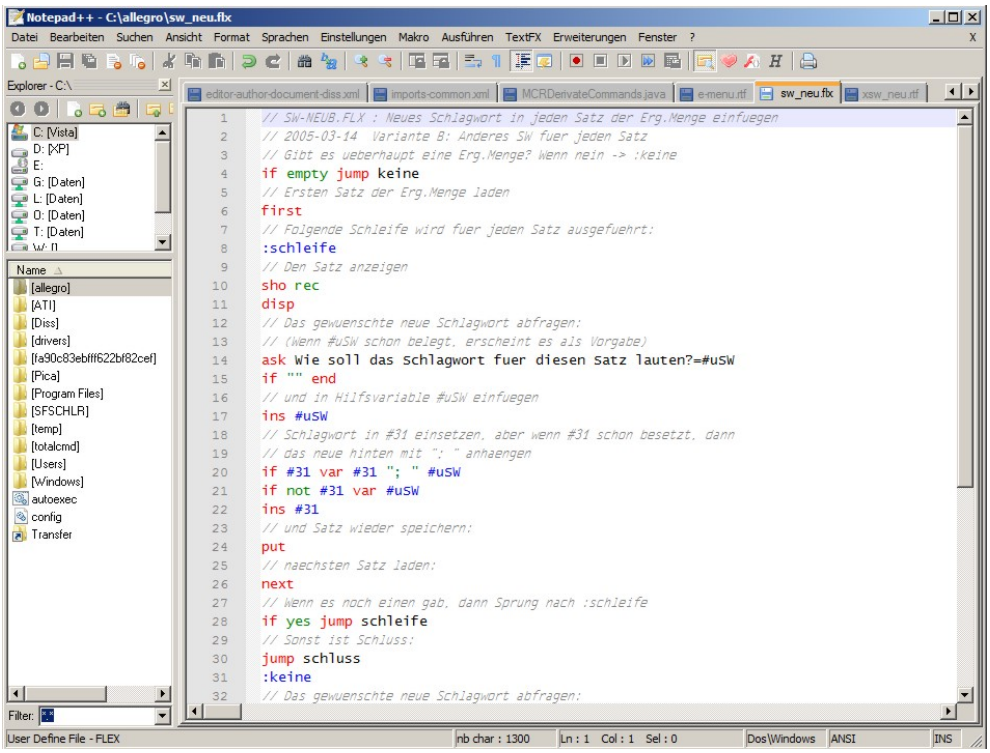

Abbildung 9: sw_neu.flx mit Notepad++ und Syntax Highlighting

Diesen *FLEX* können Sie nun als Grundlage für weitere *FLEX*e nehmen, mit denen Sie an der Ergebnismenge etwas ändern wollen. Der Hauptteil dieses *FLEX*es besteht nämlich aus den Zeilen:

```
if #31 var #31 "; " #uSW
if not #31 var #uSW
insert #31
//  und Satz wieder speichern:
put
```

Laden Sie diesen *FLEX* in einen Editor, bearbeiten Sie ihn, passen Sie eventuell den Abfragetext an und speichern Sie ihn unter einem neuen Namen ab. Schon haben Sie einen weiteren *FLEX* erstellt, der Änderungen in der Ergebnismenge vornehmen kann.

Vielleicht möchten Sie den *FLEX* auch auf einen der 8 FLIP Buttons legen?

Wenn Sie zum Beispiel nur für die Dauer Ihrer aktuellen *allegro* Sitzung Ihren *FLEX* auf den Button 1 legen möchten, geben Sie folgendes im Schreibfeld ein:

```
x flip 1&1: SW_NEU= X sw_neu
```

Danach können Sie mit der Tastenkombination ALT+1 den *FLEX* starten. Bei der nächsten Sitzung ist er allerdings wieder weg!
Soll der *FLEX* dauerhaft auf einen der FLIP Buttons gelegt werden, so müssen Sie die bereits kennengelernte Datei _start.flx aus Kapitel 5.3 bearbeiten. Ersetzen Sie dabei einfach den *FLEX*, von dem Sie meinen, ihn nicht zu brauchen mit Ihrem eigenen.

Hilfreich ist es eventuell auch, einen *FLEX* aus einer Hilfedatei heraus zu starten. Diese Hilfedateien sind im Format *.rtf und Sie können diese mit *allegro* oder mit einem Editor Ihrer Wahl bearbeiten. Auch hierfür können wir für Notepad++ ein Syntax Highlighting zur Verfügung stellen. Zur Einbindung eines *FLEX*es in eine *.rtf Datei empfiehlt es sich aber, *allegro* zu benutzen.
Legen Sie zuerst einmal über das Füllhorn eine neue Textdatei an und schreiben Sie zum Beispiel folgenden Text hinein:

```
Eigenes Menü

Neues Schlagwort einsetzen
Jeden Datensatz der Ergebnismenge mit einem neuen Schlagwort versehen
```

Markieren Sie den Text "Neues Schlagwort einsetzen" und klicken Sie mit der rechten Maustaste darauf. Wählen Sie die Funktion 'FLIP' an. Es wird eine Zeile erscheinen, die in etwa so aussehen wird:

```
?Neues Schlagwort einsetzen=
```

Dadurch wird aus dem markierten Text ein FLIP, der die Anweisung ausführt, die hinter dem '='-Zeichen eigetragen wird. In unserem Beispiel ist dies

X *sw_neu*

Speichern Sie die neue Datei unter einem Namen Ihrer Wahl ab, mit der Einschränkung, daß sie nicht den gleichen Namen wie der *FLEX* trägt!
Nennen wir sie also einfach einmal 'xsw_neu.rtf'. Somit wissen wir anhand des Namens, daß der *FLEX* sw_neu etwas damit zu tun haben muß.
Rufen Sie diese Datei einfach einmal mit

h *xsw_neu*

im Schreibfeld auf.
Sollte der FLIP nicht funktionieren, haben sich beim Abspeichern ein paar unerwünschte Zeichen in die rtf-Datei 'gemogelt'. Öffnen Sie in diesem Fall die rtf-Datei noch einmal mit einem Editor und suchen Sie nach den Zeilen, in denen der FLIP deklariert wird. Finden sich in dieser Zeile Elemente wie **\f0** oder **\f1** dann löschen Sie diese Elemente und speichern die Datei mit den vorgenommenen Änderungen noch einmal ab, dann sollte alles funktionieren!

7 Der Editor Notepad++ (NPP) mit FLEX Erweiterung

Wie im vorangegangen Kapitel beschrieben, empfehlen wir zur Bearbeitung und/oder Erstellung von *FLEX*en den Open-Source-Editor Notepad++. Zusammen mit den von der UB Braunschweig erstellten Erweiterungen wie Syntax Highlighting (SHL) oder Auto-Vervollständigung (AV) wird '*FLEX*en' für den Anwender noch einfacher und übersichtlicher.

Die SHL und AV Erweiterungen sowie eine NPP UB-Version sind unter folgender Adresse im Internet zu finden.
http://www.allegro-c.de/flex/index.htm

Auf dieser Seite finden Sie neben der pdf-Fassung dieses Handbuchs auch die FLEX-Erweiterung für ein bereits installiertes NPP sowie ein Komplettpaket, bestehend aus NPP inklusive der von uns erstellten Erweiterungen.

7.1 Benutzung der NPP FLEX-Erweiterungen

Die *FLEX* Erweiterungen sind sowohl für bestehende *FLEX*e als auch für die Erstellung neuer FLEXe benutzbar. Bei bestehenden *FLEX*en wird automatisch das SHL aktiviert und ist sofort nutzbar. Die AV als Befehlsübersichtsliste ist über die Tastaturkombination STRG+Leerzeichen abrufbar (siehe Abbildung 10).

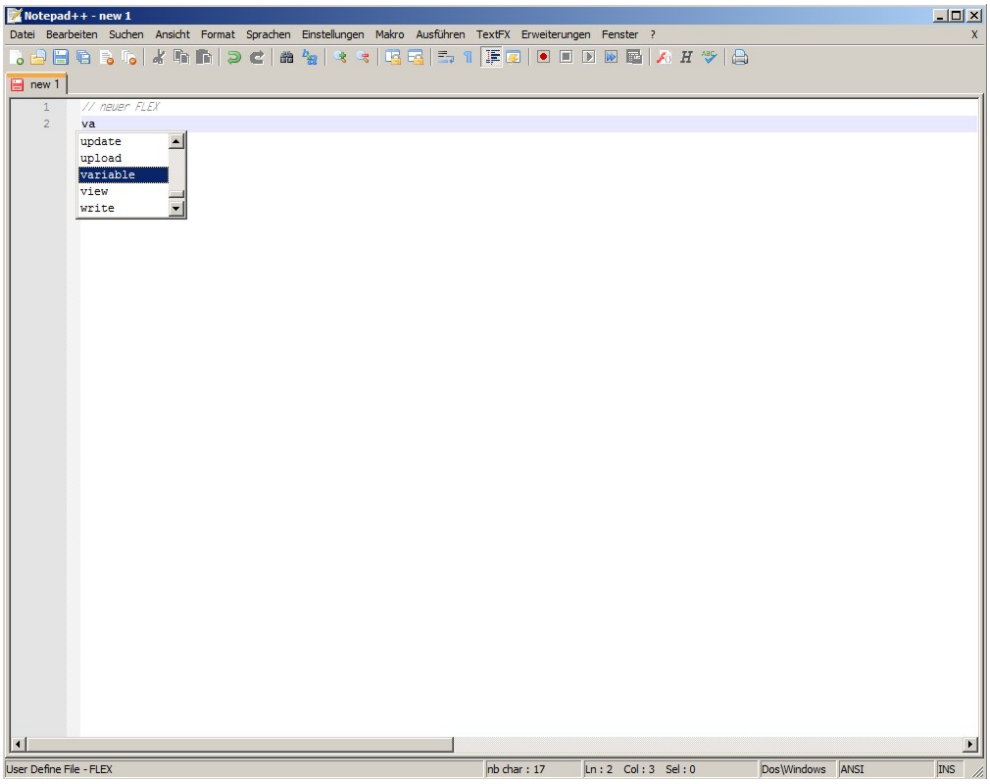

Abbildung 10: Aufruf der AV-Erweiterung unter NPP

Ist ein *FLEX*-Befehl innerhalb eines *FLEX*es bereits verwendet worden, so ist es möglich mit Hilfe der Tastenkombination aus `Shift+STRG+Leerzeichen` diesen *FLEX*-Befehl in der bereits vorwendeten Weise (kurz oder vollständig ausgeschrieben) sofort einzufügen.

7.1.1 Erstellung eines neuen *FLEX*es

Möchte man einen neuen *FLEX* erstellen und die Erweiterungen nutzen, so ist NPP mitzuteilen, daß es sich um eine *FLEX*-Datei handelt. Dies geschieht über den Menüpunkt <Sprachen> im NPP (siehe Abbildung 11)

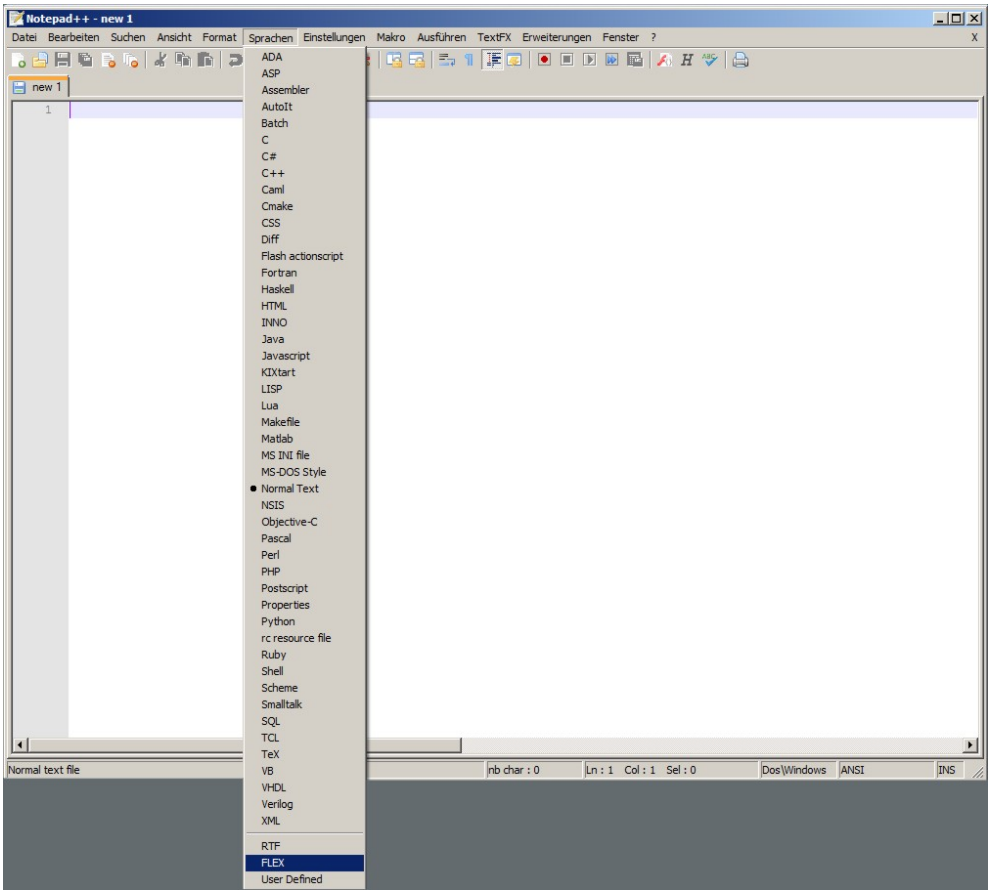

Abbildung 11: Sprachmenü von NPP

7.1.2 Vergleichen von *FLEX*-Dateien

Besteht die Notwendigkeit, *FLEX*-Dateien zu vergleichen, z.B. um zwei unterschiedliche Versionen eines *FLEX*es gegenüberzustellen, so bietet NPP auch für diesen Fall eine Lösung an. Öffnen Sie hierzu beide Dateien die Sie gegenüberstellen möchten. Diese werden jetzt als 'Tabs' im NPP angezeigt. Klicken Sie nun mit der rechten Maustaste auf den entsprechenden 'Tab' des geöffneten *FLEX*es, der in einer neuen Ansicht dargestellt werden soll, und wählen Sie den Menüpunkt '*Verschieben in eine andere Ansicht*' (siehe Abbildung 12). Alternativ können Sie diesen Menüpunkt auch unter 'Ansicht' im NPP erreichen.

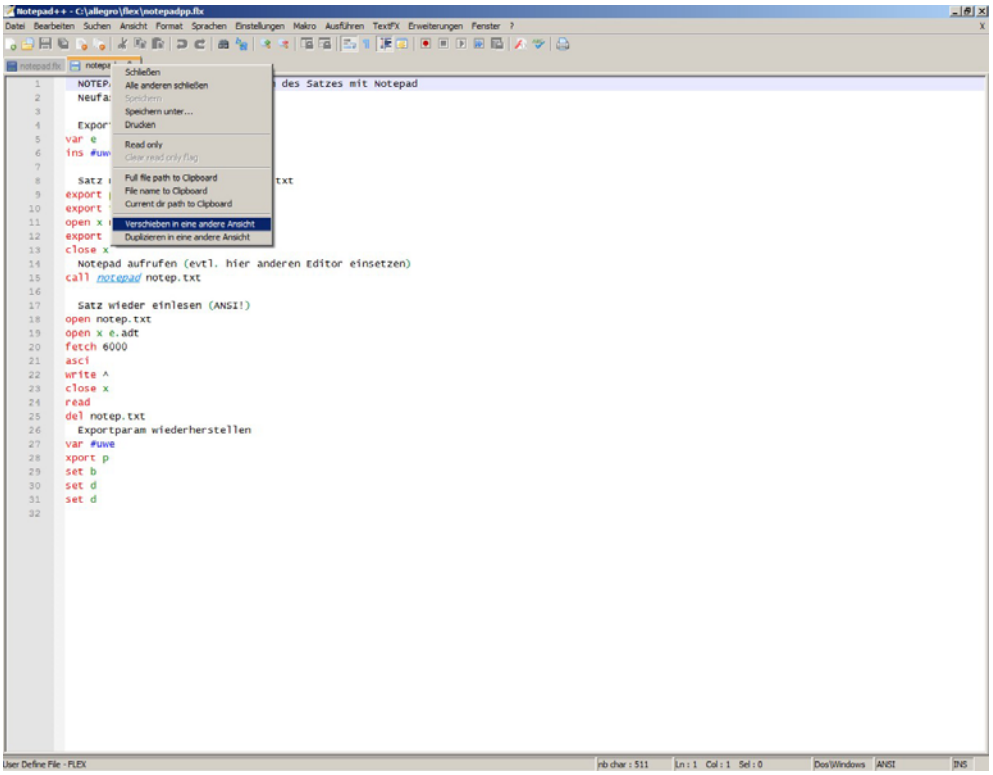

Abbildung 12: Dateivergleich mit NPP (1)

Haben Sie diese Schritte ausgeführt, so erhalten Sie die der Abbildung 13 zu entnehmende Ansicht.

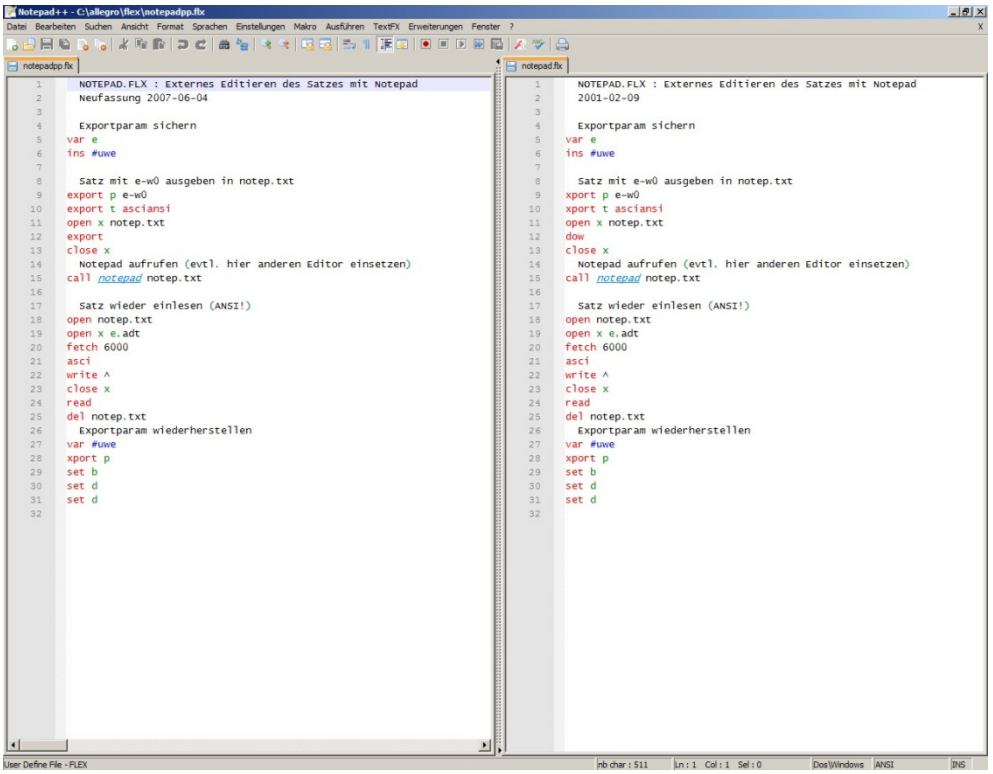

Abbildung 13: Dateivergleich mit NPP (2)

TIP: Ein weiteres Tool zum Vergleich von Dateiinhalten, jedoch ohne SHL und AV Erweiterung ist WinMerge (Zu beziehen unter http://winmerge.org/).

7.1.3 Bekannte Probleme bei der Verwendung von NPP

Ein häufiger Fehler bei der nachträglichen Verwendung der *FLEX* Add-Ons für NPP tritt auf, wenn der aktuelle Nutzer nicht als Administrator eingeloggt ist. dies liegt daran, daß die Standard-Installation der Add-Ons die notwendigen Dateien Notepad++ Verzeichnis des Administrator Userkontos ablegt. Um auch als Standard Nutzer die Erweiterungen nutzen zu können, kopieren Sie unter Windows die Datei

```
userDefineLang.xml
```

in das Verzeichnis:

```
C:\Dokumente und Einstellungen\<user>\Anwendungsdaten\Notepad++
```

Danach stehen die Erweiterungen dem entsprechenden User zur Verfügung.

8 Hilfe im *allegro* System

Sollten dennoch Fragen auftreten, so bietet Ihnen *allegro* verschiedene Optionen an, die passende Antwort auf Ihre Frage(n) zu finden. An dieser Stelle sollen Ihnen die im Zusammenhang mir *FLEX* gebräuchlichsten Wege aufgezeigt werden.

8.1 Das Füllhorn

Das Füllhorn (siehe Abbildung 14) liefert einen Gesamtüberblick über die Möglichkeiten, die Ihnen *allegro* und damit auch *FLEX* bieten. Hier finden Sie alphabetisch geordnet und farblich gekennzeichnet alle Themenbereiche des *allegro*-Systems wieder.

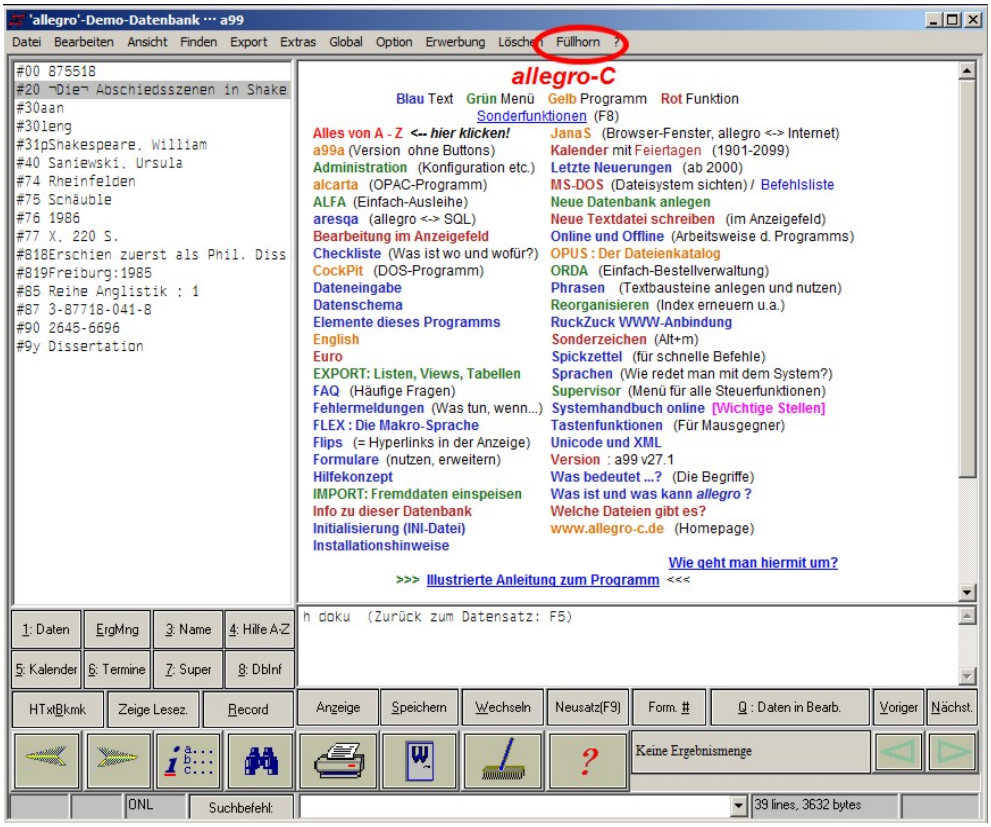

Abbildung 14: Das Füllhorn

Selbstverständlich ist hier die Dokumentation zu *FLEX* zu finden, doch auch vieles, vieles mehr. Nehmen Sie sich einfach mal die Zeit und stöbern Sie im Füllhorn, es wird sicherlich einige 'Achso'-Momente geben.

8.2 Das FLEXikon

Bestandteil des Füllhorns ist selbstverständlich auch die Dokumentation zu den *FLEX*-Befehlen und den mitgelieferten *FLEX*en, genannt *FLEX*ikon. Klicken Sie im Füllhorn auf den Menüpunkt 'FLEX: Die Makrosprache' gelangen Sie zur *FLEX* Dokumentation (siehe Abbildung 15)

Tip: Sie können dieses Seite auch erreichen, indem Sie 'h flex' im Schreibfeld eingeben

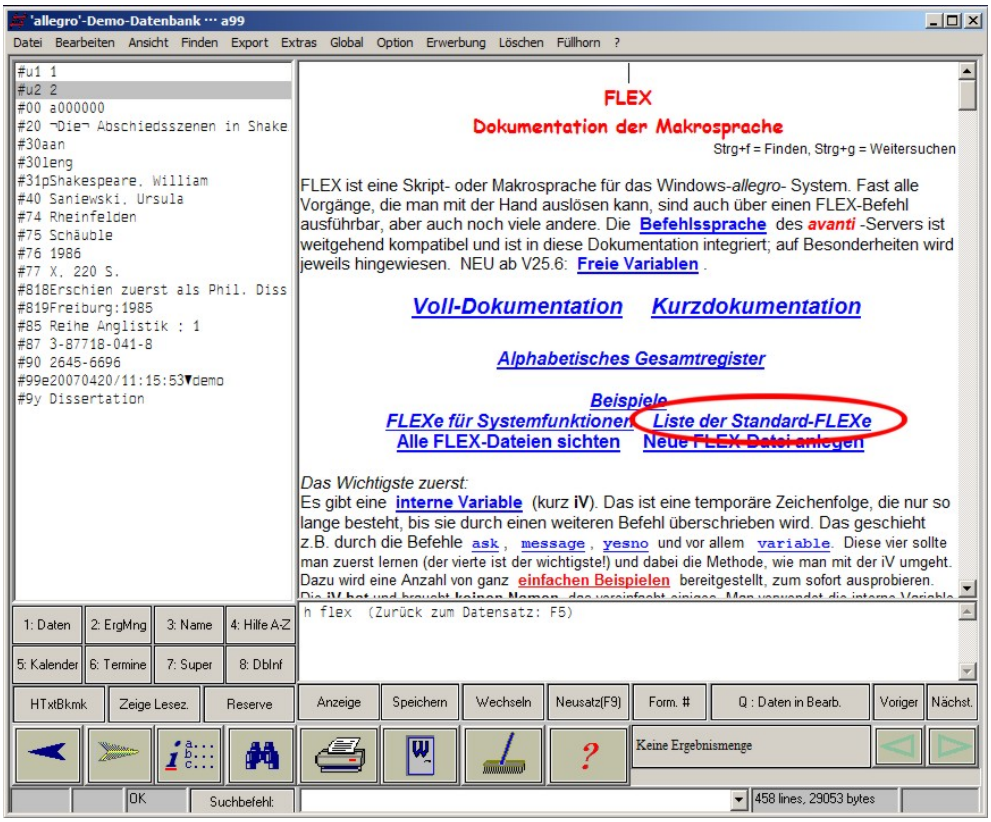

Abbildung 15: *FLEX*-Hilfeseite (Standard *FLEX*e)

Über den FLIP, der sich hinter der 'Liste der Standard-FLEXe' verbirgt, rufen Sie den *FLEX* flexikon (entspricht 'X flexikon' im Schreibfeld) auf, der Ihnen die Liste der von *allegro* mitgelieferten Standard *FLEX*e aufzeigt. Diese Liste ist nach den Einsatzgebieten der *FLEX*e geordnet und gegebenenfalls mit einem kleinen Kommentar versehen (Abbildung 16).

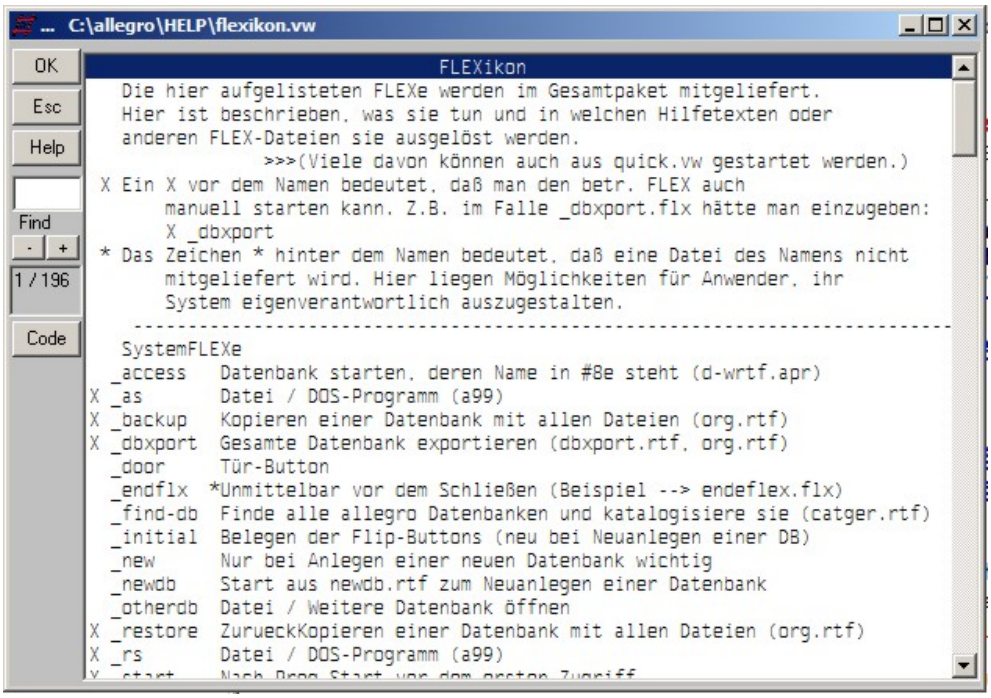

Abbildung 16: Das *FLEX*ikon

Selbstverständlich gehört die Dokumentation der *FLEX*-Befehle auch zum *FLEX*ikon. Sie können diese auch wieder über das Füllhorn unter dem Punkt 'FLEX' erreichen. Dort finden Sie das alphabetische Gesamtregister der *FLEX*-Befehle ('v flex' im Schreibfeld), sowie die Befehlsdokumentation ('h xa' im Schreibfeld) (Abbildung 17), die Ihnen bei Ihren ersten Gehversuchen mit *FLEX* mit Sicherheit hilfreich zur Seite stehen können.

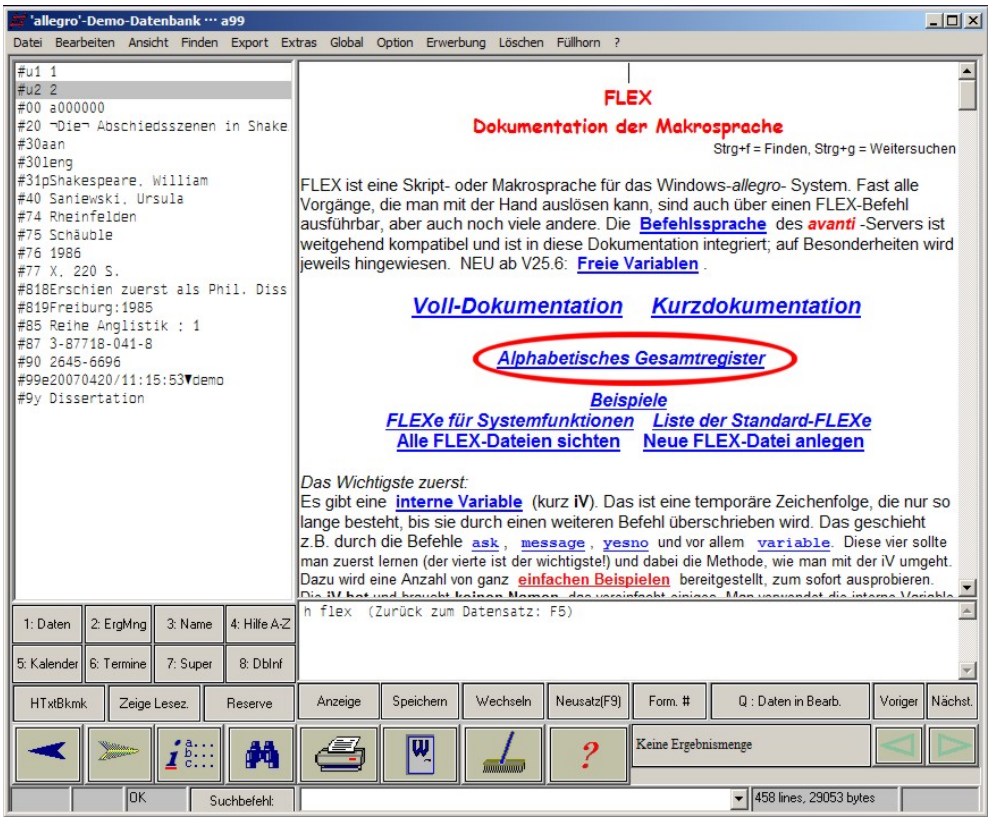

Abbildung 17: *FLEX*-Hilfeseite (*FLEX*-Befehle)

Somit sollten Sie das notwendige Rüstzeug haben, um erste Schritte in der Welt von *FLEX* tätigen zu können. Sie werden sicherlich schnell herausfinden, welche vielfältigen Möglichkeiten Ihnen *FLEX* bei Ihrem täglichen Umgang mit *allegro* bietet und wie Sie Ihre Arbeitsabläufe damit schneller und effektiver gestalten können. Wir wünschen Ihnen viel Spaß und viel Erfolg!

9 Schnellreferenz

9.1 FLEX-Befehle (thematisch)

Folgende Liste umfasst die den jeweiligen Funktionen zugeordneten *FLEX*-Befehle. Der Index '*****' bedeutet, daß es den Befehl auch in avanti gibt, der Index'**(*)**' gibt an, daß der avanti-Befehl etwas anders ist.

Funktion	***FLEX*-Befehl**	***Syntax***	***Beschreibung***
Allgemeines	include	include *dateiname*	Andere Dateien werden in einem *FLEX* an beliebiger Stelle eingebaut
	insert *	insert *#xyz*	Der Inhalt der internen Variablen iV wird in die Kategorie #xyz kopiert
	set / switch (*)	set *<Attribut>*	!: In a99 identisch, in avanti getrennte Befehle! Setzen diverser Werte für den weiteren Verlauf
	variable *	variable *cstring*	Baut beliebige Zeichenketten zusammen (siehe Kapitel 2)
Stringverarbeitung	ansi *	ansi	Wandelt den ASCII-Code der iV in ANSI-Code
	ascii *	ascii	Wandelt den ANSI-Code der iV in ASCII-Code
	crypt *	crypt	Verschlüsselt den Inhalt der iV. VERSCHLÜSSELUNG NICHT REVERSIBEL !!!
	deposit	deposit *"xABC"*	Abschnitt #-x in den Anzeigeparametern ausfüh ren (Verarbeitung intern in der iV)
	spaces	spaces	Mehrfache Leerzeichen in der iV beseitigen. Innerhalb von " " oder ' ' bleiben sie erhalten
	srx *	srx *suchbegriff*	Sucht im Text der in der iV steht nach dem angegebenen Suchbegriff
	xcode *	xcode *ab*	Kodiert den in der iV stehenden Text um
Steuerbefehle	activate	activate *i*	Das Programmfenster rückt in den Vordergrund und ist aktiv. Dabei gilt für i: 1: Auswahlfeld 2: Schreibfeld 3: Befehlszeile 4: Anzeigefeld
	call	call *<Programm>*	Aufruf eines externen Programmes
	dos/Dos	dos *Befehl*	Befehl wird in einem DOS Fenster ausgeführt
	end *	end	Der *FLEX* wird beendet
	exec	exec *flextext*	Ein anderer *FLEX* wird gestartet, der aktuelle *FLEX* wird beendet
	flex	flex *xyz*	Sendet eine Mitteilung an a99/alcarta
	if (*)	if *bedingung command*	Bedingungsprüfung
	jump *	jump *label*	Verweist auf Sprungmarke, die mit '**:**' beginnt
	perform *	perform *label*	Es wird wie bei *jump* zur Sprungmarke *:label* gesprungen. Wird der *return* Befehl erreicht, geht es mit der nach *perform* folgenden Zeile weiter
	repeat	repeat	Die Befehlskette wird so lange wiederholt, bis ein Befehl nicht ausführbar ist
	return *	return	Beendet ein laufendes Unterprogramm
	sleep	sleep *n*	Läßt das Programm n Millisekunden lang verharren
	slice	slice *n=mFLEX*	Zeitüberwachung: Eine Zeitscheibe von n Millisekunden als Timer
	STOP	STOP *programm*	Programm abrupt beenden
Rechnen / Datum	date/day	date *b*	Datum und Uhrzeit in die iV kopieren
	eval	eval *rechenbefehl*	Der Rechenbefehl wird als arithmetischer Ausdruck interpretiert und in die iV kopiert.

Funktion	*FLEX*-Befehl	*Syntax*	*Beschreibung*
Datei-I/O	addform	addform *dateiname*	Der hinter diesem Befehl angegebene Dateiname muss eine korrekte Formulardatei sein. Diese wird an das bereits geladene Formular angehängt
	close	close	Die vorher im selben (!) *FLEX* mit open zum Lesen geöffnete Datei wird zeitweilig geschlossen. Am Ende des *FLEX*es geschieht dies automatisch
	delete	delete *name*	Die hinter diesem Kommando stehende Datei wird gelöscht
	fcopy	fcopy *name1 name2*	Datei name1 wird kopiert auf name2
	fetch	fetch *number*	Aus der mit open geöffneten Datei werden number Bytes in die iV gelesen oder angehängt
	file	file *dateiname*	Der momentane Inhalt des Anzeigefeldes wird in die Datei dateiname ausgegeben
	fsize	fsize *dateiname*	Größe der Datei in Bytes wird in die iV übertragen
	ftime	ftime *dateiname*	Die Zeit der letzten Aktualisierung wird in die iV übertragen
	get (*)	get *#xyz* / get *[iV]*	Aus der mit open geöffneten Datei wird eine Zeile gelesen und in die Kategorie #xyz bzw. iV kopiert
	mkdir	mkdir *dir*	Ein neues Verzeichnis namens dir anlegen
	open	open *name*	Die Datei name wird zum Lesen geöffnet
	rename	rename *name1 name2*	Die Datei name1 wird umbenannt in name2
	write *	write	Schreibt unmittelbar in die eingestellte Exportdatei
	xml *	xml *n*	Der aktuelle Datensatz wird in einer von vier XML-Varianten ausgegeben. Dabei ist: n=0: <label>…</label> n=1: <feld nr="knum">…</feld> n=2: <label lb="label">…</label> n=3: <feld nr="knum" lb="label">…</feld>
Datenbank	***Satzebene:***		
	copy *	copy	Aktueller Datensatz wird als neue Kopie behandelt
	erase *	erase	Der aktuelle Satz wird gelöscht
	export *	export *<Parameter>*	Aktueller Satz wird exportiert
	extern	extern	Aktueller Satz wird in Datei EXX:XXX ausgegeben
	transfer	transfer *#nnn*	Kategorie #nnn aus dem Hintergrundspeicher in die iV kopieren
	input	input *nummer*	Dateinummer wird verändert
	new *	new	Ein neuer Datensatz wird angelegt
	Put / put *	Put / put	Speichern mit/ohne Rückfrage
	Ergebnismenge:		
	choose	choose *suchbefehl*	Ergebnismenge wird gebildet und angezeigt
	export *	export *<Parameter>*	Aktueller Satz wird exportiert
	load	load	Gewählter Satz aus Ergebnismenge wird geladen
	read (*)	read	Datei wird eingelesen
	family	family	Ist aktueller Satz Teil eines mehrteiligen Satzes, so wird die Ergebnismenge aus allen zusammengehörigen Sätzen gebildet
	save	save	Speichern
	order *	order *MP*	Ergebnismenge sortieren (ordnen) mit: M= Modus: a=aufsteigend d=absteigendt n=nach Satznummern P= Position des Sortierfeldes in der Kurzliste. Das erste Zeichen ist 0
	update *	update *dateiname*	Entspricht der Funktion des DOS-Update
	upload *	upload *dateiname*	Entspricht einem Update … mit Modus 01

Funktion	*FLEX*-Befehl	*Syntax*	*Beschreibung*
Datenbank	***Suche und Navigation:***		
	find (*)	find *suchbefehl*	Ergebnismenge wird gebildet
	first *	first *<Attribut>*	Der erste Satz der im Attribut angegebenen Liste.
	get (*)	get *#xyz* / get *[iV]*	Aus der mit open geöffneten Datei wird eine Zeile gelesen und in die Kategorie #xyz bzw. iV kopiert
	next *	next *<Attribut>*	Der nächste Satz der im Attribut angegebenen Liste
	prev *	prev *<Attribut>*	Der vorhergehende Satz der im Attribut angegebenen Liste
	sub	sub *#01 nnn*	Schaltet im Arbeitsspeicher zum hierarchischen Untersatz, in dessen #01 die Bandnummer nnn steht
	Index:		
	qrix (*)	qrix *N ixn abc*	Wie index, aber Kopie des Registerabschnitts ab abc in die iV
Bedienoberfläche	aresqa	aresqa *name*	Die Datei name öffnen und in einem aresqa Listenfenster anzeigen
	button	button *i*/*j*/*f*	Buttons auslösen: i: Index 1 j: Index 2 f: FIND-Button
	display	display *"xABC"*	Abschnitt #-x in den Anzeigeparamtern ausführen
	flip	flip *ixyz=flex*	Setzen von Text und *FLEX* auf den Flip-Button Nummer i(i=0…9)
	fnam	fnam *bezeichnung*/*typ*	Eine Dateiauswahlbox nach Art von Windows wird präsentiert
	help (*)	help *name*	Hilfetext wird aufgerufen
	index	index *|i abc*	Register i an der Stelle abc aufblättern
	keycheck	keycheck	Die Tastatur wird gecheckt
	last *	last *<Attribut>*	Der letzte Satz der Ergebnismenge wird geladen
	menu	menu *text*	Der Menüpunkt zwischen 'Option' und 'Löschen' wird auf den Wert Text gesetzt
	message	message *text*	Text wird in einer Messagebox angezeigt und muß mit [OK] bestätigt werden
	noyes	noyes *frage*	Frage wird in einer Ja/Nein/Abbruch-Box gezeigt. Default-Wert ist NO
	phrase	phrase *i text*	Phrasen werden belegt. i kann eine Zahl > 0 sein
	show	show *xxx*	Steuerung des Auswahlfeldes und der Anzeige. Für xxx sind diverse Möglichkeiten vorgesehen
	undo	undo	Entspricht dem Button [Wechseln]
	view	view *name*	Die ViewListe name wird geöffnet und angezeigt
	yesno	yesno *frage*	Frage wird in einer Ja/Nein/Abbruch-Box gezeigt. Default-Wert ist YES
Erfassung / Ausgabe	ask	ask *|iprompt=Vorgabe*	Aufforderung zu einer Eingabe
	ccopy/cpaste	ccopy / cpaste	Der Inhalt der iV wird in die Zwischenablage kopiert bzw. die Zwischenablage in die iV
	form	form *i* / form *titel*	Formular i wird aufgeblättert bzw. Formular mit Überschrift titel
	print	print	Anzeigefenster ausdrucken
	select	select *prompt=antw1…*	Auswahlliste anbieten mit vorgegebenen Antworten

Tabelle 1: *FLEX*-Befehle (thematisch)

9.2 FLEX-Befehle (alphabetisch)

Folgende Liste enthält die *FLEX*-Befehle in alphabetischer Reihenfolge. Auch hier haben die Indizes '*****' und '**(*)**' die gleiche Bedeutung wie in Kapitel 9.1.

***FLEX*-Befehl**	***Syntax***	***Beschreibung***	***Funktion***
A			
activate	activate *i*	Das Programmfenster rückt in den Vordergrund und ist aktiv. Dabei gilt für i: 1: Auswahlfeld 2: Schreibfeld 3: Befehlszeile 4: Anzeigefeld	Steuerbefehle
addform	addform *dateiname*	Der hinter diesem Befehl angegebene Dateiname muss eine korrekte Formulardatei sein. Diese wird an das bereits geladene Formular angehängt	Datei I/O
ansi *	ansi	Wandelt den ASCII-Code der iV in ANSI-Code	Stringverarbeitung
aresqa	aresqa *name*	Die Datei name öffnen und in einem aresqa Listenfenster anzeigen	Bedienoberfläche
ascii *	ascii	Wandelt den ANSI-Code der iV in ASCII-Code	Stringverarbeitung
ask	ask *|iprompt=Vorgabe*	Aufforderung zu einer Eingabe	Erfassung / Ausgabe
B			
button	button *i|j|f*	Buttons auslösen: i: Index 1 j: Index 2 f: FIND-Button	Bedienoberfläche
C			
call	call *<Programm>*	Aufruf eines externen Programmes	Steuerbefehle
ccopy/cpaste	ccopy / cpaste	Der Inhalt der iV wird in die Zwischenablage kopiert bzw. die Zwischenablage in die iV	Erfassung / Ausgabe
choose	choose *suchbefehl*	Ergebnismenge wird gebildet und angezeigt	Datenbank
close	close	Die vorher im selben (!) *FLEX* mit open zum Lesen geöffnete Datei wird zeitweilig geschlossen. Am Ende des *FLEX*es geschieht dies automatisch	Datei I/O
copy *	copy	Aktueller Datensatz wird als neue Kopie behandelt	Datenbank
crypt *	crypt	Verschlüsselt den Inhalt der iV. VERSCHLÜSSELUNG NICHT REVERSIBEL !!!	Stringverarbeitung
D			
date/day	date *b*	Datum und Uhrzeit in die iV kopieren	Rechnen / Datum
delete	delete *name*	Die hinter diesem Kommando stehende Datei wird gelöscht	Datei I/O
deposit	deposit *"xABC"*	Abschnitt #-x in den Anzeigeparametern ausführen (Anzeige ändert sich nicht, Verarbeitung intern in der iV)	Stringverarbeitung
display	display *"xABC"*	Abschnitt #-x in den Anzeigeparamtern ausführen	Bedienoberfläche
dos/Dos	dos *Befehl*	Befehl wird in einem DOS Fenster ausgeführt	Steuerbefehle

FLEX-Befehl	*Syntax*	*Beschreibung*	*Funktion*
E			
end *	end	Der *FLEX* wird beendet	Steuerbefehle
erase *	erase	Der aktuelle Satz wird gelöscht	Datenbank
eval	eval *rechenbefehl*	Der Rechenbefehl wird als arithmetischer Ausdruck interpretiert und in die iV kopiert.	Rechnen / Datum
exec	exec *flextext*	Ein anderer *FLEX* wird gestartet, der aktuelle *FLEX* wird beendet	Steuerbefehle
export *	export *<Parameter>*	Aktueller Satz wird exportiert	Datenbank
extern	extern	Aktueller Satz wird in Datei EXX:XXX ausgegeben	Datenbank
F			
family	family	Ist aktueller Satz Teil eines mehrteiligen Satzes, so wird die Ergebnismenge aus allen zusammengehörigen Sätzen gebildet	Datenbank
fcopy	fcopy *name1 name2*	Datei name1 wird kopiert auf name2	Datei I/O
fetch	fetch *number*	Aus der mit open geöffneten Datei werden *number* Bytes in die iV gelesen oder angehängt	Datei I/O
file	file *dateiname*	Der momentane Inhalt des Anzeigefeldes wird in die Datei dateiname ausgegeben	Datei I/O
find (*)	find *suchbefehl*	Ergebnismenge wird gebildet	Datenbank
first *	first *<Attribut>*	Der erste Satz der im Attribut angegebenen Liste.	Datenbank
flex	flex *xyz*	Sendet eine Mitteilung an a99/alcarta	Steuerbefehle
flip	flip *ixyz=flex*	Setzen von Text und *FLEX* auf den Flip-Button Nummer i(i=0…9)	Bedienoberfläche
fnam	fnam *bezeichnung*/*typ*	Eine Dateiauswahlbox nach Art von Windows wird präsentiert	Bedienoberfläche
form	form *i* / form *titel*	Formular i wird aufgeblättert bzw. Formular mit Überschrift titel	Erfassung / Ausgabe
fsize	fsize *dateiname*	Größe der Datei in Bytes wird in die iV übertragen	Datei I/O
ftime	ftime *dateiname*	Die Zeit der letzten Aktualisierung wird in die iV übertragen	Datei I/O
G			
get (*)	get *#xyz* / get *[iV]*	Aus der mit open geöffneten Datei wird eine Zeile gelesen und in die Kategorie #xyz bzw. iV kopiert	Datei I/O; Datenbank
H			
help (*)	help *name*	Hilfetext wird aufgerufen	Bedienoberfläche
I			
if (*)	if *bedingung command*	Bedingungsprüfung	Steuerbefehle
include	include *dateiname*	Andere Dateien werden in einem *FLEX* an beliebiger Stelle eingebaut	Allgemeines
index	index *|i abc*	Register i an der Stelle abc aufblättern	Bedienoberfläche
input	input *nummer*	Dateinummer wird verändert	Datenbank
insert *	insert *#xyz*	Der Inhalt der internen Variablen iV wird in die Kategorie #xyz kopiert	Allgemeines
J			
jump *	jump *label*	Verweist auf Sprungmarke, die mit ':' beginnt	Steuerbefehle
K			
keycheck	keycheck	Die Tastatur wird gecheckt	Bedienoberfläche

FLEX-Befehl	*Syntax*	*Beschreibung*	*Funktion*
L			
last *	last *<Attribut>*	Der letzte Satz der Ergebnismenge wird geladen	Bedienoberfläche
load	load	Gewählter Satz aus Ergebnismenge wird geladen	Datenbank
M			
menu	menu *text*	Der Menüpunkt zwischen 'Option' und 'Löschen' wird auf den Wert Text gesetzt	Bedienoberfläche
message	message *text*	Text wird in einer Messagebox angezeigt und muß mit [OK] bestätigt werden	Bedienoberfläche
mkdir	mkdir *dir*	Ein neues Verzeichnis namens dir anlegen	Datei I/O
N			
new *	new	Ein neuer Datensatz wird angelegt	Datenbank
next *	next *<Attribut>*	Der nächste Satz der im Attribut angegebenen Liste	Datenbank
noyes	noyes *frage*	Frage wird in einer Ja/Nein/Abbruch-Box gezeigt. Default-Wert ist NO	Bedienoberfläche
O			
open	open *name*	Die Datei name wird zum Lesen geöffnet	Datei I/O
order *	order *MP*	Ergebnismenge sortieren (ordnen) mit: M= Modus: a=aufsteigend d=absteigendt n=nach Satznummern P= Position des Sortierfeldes in der Kurzliste. Das erste Zeichen ist 0	Datenbank
P			
perform *	perform *label*	Es wird wie bei *jump* zur Sprungmarke *:label* gesprungen. Wird der *return* Befehl erreicht, geht es mit der nach *perform* folgenden Zeile weiter	Steuerbefehle
phrase	phrase *i text*	Phrasen werden belegt. i kann eine Zahl > 0 sein	Bedienoberfläche
prev *	prev *<Attribut>*	Der vorhergehende Satz der im Attribut angegebenen Liste	Datenbank
print	print	Anzeigefenster ausdrucken	Erfassung / Ausgabe
Put / put *	Put / put	Speichern mit/ohne Rückfrage	Datenbank
Q			
qrix (*)	qrix *N ixn abc*	Wie index, aber Kopie des Registerabschnitts ab abc in die iV	Datenbank
R			
read (*)	read	Datei wird eingelesen	Datenbank
rename	rename *name1 name2*	Die Datei name1 wird umbenannt in name2	Datei I/O
repeat	repeat	Die Befehlskette wird so lange wiederholt, bis ein Befehl nicht ausführbar ist	Steuerbefehle
return *	return	Beendet ein laufendes Unterprogramm	Steuerbefehle

FLEX-Befehl	*Syntax*	*Beschreibung*	*Funktion*
S			
save	save	Speichern	Datenbank
select	select *prompt=antw1…*	Auswahlliste anbieten mit vorgegebenen Antworten	Erfassung / Ausgabe
set / switch (*)	set *<Attribut>*	!: In a99 identisch, in avanti getrennte Befehle! Setzen diverser Werte für den weiteren Verlauf	Allgemeines
show	show *xxx*	Steuerung des Auswahlfeldes und der Anzeige. Für xxx sind diverse Möglichkeiten vorgesehen	Bedienoberfläche
sleep	sleep *n*	Läßt das Programm n Millisekunden lang verharren	Steuerbefehle
slice	slice *n=mFLEX*	Zeitüberwachung: Eine Zeitscheibe von n Millisekunden als Timer	Steuerbefehle
spaces	spaces	Mehrfache Leerzeichen in der iV beseitigen. Innerhalb von " " oder ' ' bleiben sie erhalten	Stringverarbeitung
srx *	srx *suchbegriff*	Sucht im Text der in der iV steht nach dem angegebenen Suchbegriff	Stringverarbeitung
STOP	STOP *programm*	Programm abrupt beenden	Steuerbefehle
sub	sub *#01 nnn*	Schaltet im Arbeitsspeicher zum hierarchischen Untersatz, in dessen #01 die Bandnummer nnn steht	Datenbank
T			
transfer	transfer *#nnn*	Kategorie #nnn aus dem Hintergrundspeicher in die iV kopieren	Datenbank
U			
undo	undo	Entspricht dem Button [Wechseln]	Bedienoberfläche
update *	update *dateiname*	Entspricht der Funktion des DOS-Update	Datenbank
upload *	upload *dateiname*	Entspricht einem Update … mit Modus 01	Datenbank
V			
variable *	variable *cstring*	Baut beliebige Zeichenketten zusammen (siehe Kapitel 2)	Allgemeines
view	view *name*	Die ViewListe name wird geöffnet und angezeigt	Bedienoberfläche
W			
write *	write	Schreibt unmittelbar in die eingestellte Exportdatei	Datei I/O
X			
xcode *	xcode *ab*	Kodiert den in der iV stehenden Text um	Stringverarbeitung
xml *	xml *n*	Der aktuelle Datensatz wird in einer von vier XML-Varianten ausgegeben. Dabei ist: n=0: <label>…</label> n=1: <feld nr="knum">…</feld> n=2: <label lb="label">…</label> n=3: <feld nr="knum" lb="label">…</feld>	Datei I/O
Y			
yesno	yesno *frage*	Frage wird in einer Ja/Nein/Abbruch-Box gezeigt. Default-Wert ist YES	Bedienoberfläche

Tabelle 2: *FLEX*-Befehle (alphabetisch)

9.3 Die internen Sondervariablen

Der folgenden Übersicht sind die in Kapitel 3.2 beschriebenen internen Sondervariablen (iS) sowie eine kurze Beschreibung und ein Ausgabebeispiel zu entnehmen.

iS	*Beschreibung*	*Ausgabebeispiel*
A	Access-Modus der aktuellen Sitzung (Berechtigung aus INI-Datei)	31
B	Name der Datenbank	cat
C	Copy-Verzeichnis (für Sicherungskopie, siehe _BACKUP.FLX / _RESTORE.FLX)	
D	Datenbank Pfadname (mit \ am Ende)	C:\allegro\demo2\
E	Name der aktuellen Exportdatei	test.txt
Err	Fehlermeldungstext	
F	File - dahinter folgt direkt ein Dateiname; der *Inhalt* dieser Datei wird in die Ausgabe eingefügt; folgt kein Dateiname, wird Inhalt der iV als Name genommen. Steuerzeichen bleiben dabei erhalten.	
G	Name der LOG Datei (falls nicht default C:\allegro\demo2\cat.log)	
H	Liste der Register Überschriften	
I	Indexliste (Die Liste der symbolischen Registernamen PER 1 Personennamen | TIT 3 Titelwörter…	
J	Feldname laut CFG zu der Feldnummer, die gerade in der iV steht	
Jl	Länge der iV, d.h. der momentane Inhalt der iV wird durch deren Länge ersetzt	
Jlx **JRx**	Inhalt der iV nach Links oder Rechts durch das Zeichen x auf soviele Stellen auffüllen, wie es der interne Zähle angibt	
K	Name der Konfigurationsdatei	a
K1	Nur der erste Buchstabe der Konfigurationsdatei	a
Kk	Werte k und t aus der CFG	2/4
L	Sprachenbezeichnung (INI Befehl: language)	Ger
M	Inhalt der Environmentvariable TEMP	C:\<Local-PATH>\Temp\
O	Name der Offline Datei	
P	Name des Programmverzeichnisses	C:\allegro\
Q	Fokusziffer (gibt an, wo sich momentan der Cursor befindet) 1: Auswahlfeld 2: Schreibfeld 3: Befehlszeile 4: Anzeigefeld	4
R	Restriktionen: Liste der Namen mit Restriktionen (R-Befehle) PYR Erscheinungsjahr TYP Dokument BDT Bestelldaten	
S	Short-Title Überschrift	Titel ^Verf ^Jahr^Signatur
T	Titel der Datenbank	'*allegro*'-Demo Datenbank
U	Name der aktuellen Ergebnismenge	
Uv	Name der zuletzt benutzten View-Liste	
V	Name der INI-Datei (Vorgaben), mit der gestartet wurde	C:\allegro\demo2\orda.ini
W	Arbeitsverzeichnis (Working directory)	C:\allegro\
X	Name der letzten Hilfedatei	xcstring
Y	Pfad der Indexdatei (InxName in INI, falls anders als D, sonst leer)	
Z	Wert der internen Zahlenvariablen iZ	1
Zk	Wert der iZ gerundet auf k Stellen nach dem Komma (k=0…9)	1.0000

iS	*Beschreibung*	*Ausgabebeispiel*
a	Anzeige der aktuellen Anzeigeparameter	d-wrtf
b	Breite der Kategoriennummern, Textumfang. Bei A.CFG:2,4	2,4
ci	Zeile i der UIF Datei. z.B. C339	"Datenbank-Information"
ca	Arbeitsspeicher Belegung / Max.Größe // Anzahl Felder,MaxZahl	353/30000//17,250
cr	Reservespeicher Belegung / Max Größe // Anzahl Felder,MaxZahl	797/36000//48,600
cp	Phrasenspeicher Belegung / Max. Größe // Anzahl Felder, MaxZahl	6996/16000//473,1200
d	Name der aktuellen Druckparameter	p-w
e	Name der aktuellen Exportparameter	e-w
f	find Befehl (Name) der letzten ErgMenge	|9 db ?
g	Zeile im Index, die zuletzt benutzt wurde	
h	Headline des Registers, dessen Nummer in der iZ steht	
i	interne Nummer des Datensatzes	293
j	Nummer der cLD-Datei des Satzes	1
jb	Länge des Datensatzes in byte (Aktuelle Länge im Arbeitsspeicher)	336
jf	Länge des Datensatzes in der Datei	336
jp	Position des Satzes in der Datei	
k0	Anzahl der Kategorien (Felder) des aktuellen Satzes	17
k_1	Erstes Feld des aktuellen Satzes	00 875518
k_2	Nächstes Feld (leer, wenn letztes erreicht war)	20 Die Abschiedsszenen…
k_3	Letztes Feld	9y Dissertationen
k_4	Voriges Feld (leer, wenn erstes erreicht war)	20 Die Abschiedsszenen…
kn	Gesamter Datensatz, Felder mit # beginnend und Code '10' als Feldende	
kr	Wie kn, aber mit '13 10' statt '10'	
l	Größe (length) der Ergebnismenge	0
m	Name des Programms und Nummer der Version	a99 v27.1
n	neue Zeile, gleichwertig zu '13 10'	
o	Anzahl der Offline-Sätze	0
p	Primärschlüssel des aktuellen Satzes	|9875518
q	Nummer der aktuellen Ergebnismenge	3/2
r	relative Nummer des Satzes in der Ergebnismenge	0
s	Kurzzeile des Satzes (aus der STL-Datei)	Expl.Satz /2520-1295
sK	Alle Registereinträge zum aktuellen Satz, getrennt durch ==	
sk	Wie sK, jedoch getrennt durch '13 10' (Zeilenvorschub) - Listenansicht	
t	Gesamtzahl der Sätze (total number of records)	900
u	Datum und Uhrzeit	20070402/12:15:36
v	Phrasendatei	demophr.a99
vi	Phrase i (i=1……255) (->phrase i text)	
w	Inhalt des Schreibfeldes	x var w \mes
x	Name des externen Editors	x
y	Name der Indexparameter (InxParam in INI, falls anders als B, sonst B)	cat
z	Wert des internen Zählers	0
z0	Anzahl Zeilen in der Anzeige (inkl. aller Leerzeilen)	194
z1	Erste Zeile des Anzeigefeldes	
z2	Nächste Zeile	Register
z3	Letzte Zeile (Leerzeile am Ende unberücksichtigt)	Satztyp Dissertation
z4	Vorherige Zeile	
zc	Inhalt der aktuellen Zeile	
zn	Nummer der aktuellen Zeile	69
zv	Nummer der oben sichtbaren Zeile	43

Tabelle 3: interne Sondervariablen

9.4 Vorhandene FLEXe

Neben der Möglichkeit eigene *FLEX*e zu erstellen, stellt das *allegro* System selbstverständlich auch fertige *FLEX*e zur Verfügung. Ein Auszug davon ist in alphabetischer Reihenfolge der folgenden Liste zu entnehmen:

FLEX	*manuell startbar*	*Beschreibung*
System-FLEXe:	*Beginnend mit '_'*	
_access		Datenbank starten, deren Name in #8e steht (d-wrtf.apr)
_as	X	Wird in ALCARTA vom Menü 'Datei | DOS-Programm' aktiviert / Ruft APAC auf
_backup	X	Kopieren einer Datenbank mit allen Dateien (org.rtf)
_dbxport	X	Gesamte Datenbank exportieren (dbxport.rtf, org.rtf)
_door		Wird durch den Button mit der Tür aktiviert. Ruft door.bat auf und setzt die dazu notwendigen Optionen
_endflx		Wird unmittelbar vor dem Schließen der Datenbank ausgelöst.
_find-db		Finde alle *allegro* Datenbanken und katalogisiere sie (catger.rtf)
_initial		Belegen der FLIP Buttons bei Neuanlegung einer Datenbank
_new		Wird gestartet wenn bei noch leerer Datenbank 'Index' gedrückt wird. Die Erfassung des ersten Datensatzes wird ausgelöst
_newdb		Wird zum Neuanlegen einer Datenbank aus newdb.rtf heraus gestartet
_otherdb		Weitere Datenbank öffnen
_restore	X	Rücksicherung einer Datenbank mit allen Dateien (org.rtf)
_rs	X	Datei / DOS Programm (a99)
_start	X	Wird unmittelbar nach Programmstart und nach 'password.flx' ausgelöst. Optimaler Ort, um die FLIP Buttons zu belegen
aLF FLEXe:	*nur für aLF Anwender*	
alf		Ausleihfunktion initialisieren
a-exemp	X	Neuen Exemplarsatz anlegen
a-trenn	X	Hierarchische Sätze in verknüpfte Sätze zerlegen
a-*		Weitere Befehle, NUR für aLF Anwender
ald-chk	X	ALD Datei auf Konsistenz prüfen
ald-erg	X	ALD Datei als Ergebnismenge präsentieren
aparedit	X	Anzeigeparameter bearbeiten (adm.rtf)
Aresqa		
aq	X	Tabelle mit Aresqa bearbeiten
aresqa1		Tabelle und SQL-Skript produzieren (table.rtf)
aresqa2		Einlesen einer Tabelle (vorher mit SQL bearbeitet) (table.rtf)
ausl		ALFA Ausleihfunktion (d.wrtf.apr)
bedit		Bequemes externes Editieren eines Datenfeldes (onerror.flx)
bfile	X	Binäre Datei lesen, schreiben und manipulieren
bookmark		Lesezeichen für Hilfetexte als Viewliste verwalten
cfga	X	Datenfelder und –teilfelder mit Klartextbezeichnung anzeigen
cfgb	X	Anzeigefeld-Inhalt in den Datensatz überführen
cfgedit	X	CFG-Datei editieren
check-e	X	Leersätze checken: Register 1.Einträge mit //
cockpit	X	CockPit starten
copy		Für Schnellkopplung notwendig: kopierten Satz einlesen
cstring	X	Werte aller *FLEX* Sondervariablen zeigen
Für Zählungen:	*siehe ct.rtf*	
ctpick		Wählt eine vorhandene C-*.xCT Datei aus, damit diese später verwendet werden kann
ctprod		Wertet die ausgewählte .ACT aus
ctstart		Ausgelöst durch 'ALT+9': Kopiert die Anzeige in Datei C-*.xCT, prüft vorher den Namen *

FLEX	*manuell startbar*	*Beschreibung*
dbinfo	X	Info zur Datenbank zusammenstellen und anzeigen (doku.rtf)
dcinput	X	Eingabe neuer Datensatz mit DublinCore-Metadaten
dir		Anzeige der Dateiliste eines Verzeichnisses (aus onerror.flx)
dirs		Gesamtliste aller Dateien und Unterverzeichnisse eines Ordners
dispedit	X	Bearbeitung des Datensatzes im Anzeigefeld vorbereiten
dispread		Bearbeiteten Datensatz aus der Anzeige in den Arbeitsspeicher kopieren
dollar	X	Umrechnung Dollar <-> Euro (Füllhorn = doku.rtf)
dos-cmd		Bestimmten DOS-Befehl demonstrieren (aus dos-cmd.rtf)
dos	X	Auswahlliste der wichtigen Verzeichnisse. Start von DOS Fenstern
dosfile		Dateri auswählen, danach dos.flx starten
doswin		DOS-Fenster aus bestimmten Verzeichnis öffnen (aus dos.flx)
dparedit	X	Display Parameter bearbeiten (adm.rtf)
endeflex		Beispiel für einen _endflex
euro	X	Umrechnung Euro <-> Dollar (Füllhorn = doku.rtf)
file-zer	X	Texdatei in 26 Dateien zerlegen. Sortiert nach den Anfangsbuchstaben der Zeilen
file	X	ASCII-Datei zeilenweise lesen und schreiben und dabei verändern
files	X	Interaktive Dokumentation der Dateitypen
flexikon	X	Die thematische Übersicht der Befehle
fnam	X	Verzeichnisse zur Auswahl anzeigen / umschalten
formedit	X	Formulardatei bearbeiten (adm.rtf)
fremd		Vorliegende Fremddatei importieren
fremddb	X	Schnellkopplung: Weitere Datenbank zur Datenübernahme starten
ftf	X	Volltextsuche in der gesamten Datenbank per *FLEX*
fts		Full Text Search. Ergebnisse dann als a99 Ergebnismenge
gauss	X	Osterdatum berechnen
helpedit	X	Bearbeiten der datenbankspezifischen Hilfedatei
hilite	X	Syntax-Highlight-Anzeige eine Export-Parameterdatei (hilite.rtf)
holidays	X	Feiertagsliste produzieren (Aufruf aus kalender.flx)
htmform		Beispiel für HTML-Formular für JanaS
iniedit	X	INI-Dateien bearbeiten (adm.rtf)
input		Auswahl des Satztyps. Start des Formulars (aus input.vw)
iparedit	X	Indexparameterdatei bearbeiten (adm.rtf)
kalender	X	Kalenderblatt produzieren und anzeigen (Füllhorn)
language	X	Umschalten auf andere Sprache (Englisch <-> Deutsch) (Füllhorn)
leapyear	X	Feststellen, ob eingegebenes Jahr ein Schaltjahr ist
leer	X	Neue Textdatei im Anzeigefeld eingeben
log	X	Aufbereitung und Anzeige der LOG-Datei
logcheck		Registrieren, ob jemand etwas eingibt (slice-Prozedur)
memo		Finde aktuelle Termine (slice-Prozedur) (memo.rtf)
memodate		Liste der Memos oder Termine anzeigen
mini		Anlegen einer Kopie einer Mini-Datenbank (3 Typen) (mini.rtf)
nel	X	Ergebnismenge für Neuerwerbsliste bilden (nur für $A.CFG)
nextnum		Unterprogramm zum Vergeben der nächsten laufenden Nummer
notepad	X	Externes Editieren des aktuellen Satzes mit Notepad
npw	X	Neues Passwort an einen Nutzer vergeben
ORDER-FLEXe: *z.T. nur für ORDER Anwender, einige auch für ORDA*		
o-*		siehe order.rtf und orda.rtf

FLEX	*manuell startbar*	*Beschreibung*
On-FLEXe:	*(werden automatisch in bestimmten Situationen aktiviert. Ist einer dieser FLEXe nicht vorhanden, wird eine 'default'-Aktion ausgelöst, weshalb nicht alle diese FLEXe mitgeliefert werden *)*	
oncopy		Wird beim Anlegen eines neuen Datensatzes ausgeführt, wenn dieser als Kopie eines anderen besteht und oninpout.flx nicht existiert
ondfile*		Menüpunkt 'Datei / Anzeige speichern als…', muss den Befehl *file* enthalten
onerase*		Menüpunkt 'Löschen' eines Satzes. Wird der Wert *NO* in die iV geschrieben, wird das Löschen nicht ausgeführt
onerror		Wird ausgeführt, wenn im Schreibfeld eine falsche Eingabe erfolgt ist
onexprec*		Export des aktuellen Satzes
onexpset*		Export der aktuellen Ergebnismenge
onextern		Entspricht '*ALT+t*' – Externes Editieren
onf2		Wenn *F2* gedrückt wird
onf3/onf4*		siehe onf2
onf8		Wenn F8 gedrückt wird. Wenn nicht vorhanden: Hintergrundspeicher -> Anzeige
onf11*		Wenn nicht vorhanden: Anzeige umschalten breit <-> schmal
onf12*		Wenn nicht vorhanden: Hilfeseite Tastenbelegung
onfnda99		Wenn in der Suchbefehlszeile etwas falsches eingegeben wurde (a99)
onfndalc*		siehe onfnda99, hier für alcarta
onforms		Button Formulare
onglobal		Menüpunkt 'Global: In die iV werden die Werte '1','2' oder '3' geschrieben, je nach Wahl des Menüpunktes
onhlpa99*		Das 'Füllhorn' in a99
onhlpalc*		Das 'Füllhorn' in alcarta
oninput		Button 'Neusatz' oder F9
onnew		Wird ausgeführt, wenn ein neuer Datensatz angelegt werden soll, aber oninput.flx nicht existiert
onoffl*		Datei / weitere Offline Datei laden
onprint		Printer Button
onprop*		Menüpunkt 'Eigenschaften' des Kontextmenüs im Eingabefeld
onput		Button 'Speichern'
onexprtf*		Export / andere Exportdatei
onexprtp*		Export / andere Exportparameter
opus3	X	Start der OPUS Datenbank
org		Für CockPit-Menü 'Organisieren' (org.rtf)
ostertag	X	Ostertermine 1700 bis 2199 ausrechnen und als Liste anzeigen
param		Liste von vorhandenen Parameterdateien als ViewListe anzeigen
password		Passwort des Nutzers abfragen. Wird auf _psw.flx kopiert
print-r		Ergebnismenge mittels der Druckparameter drucken (aus print.vw)
qrix	X	Registerauszug ins Anzeigefenster oder in eine Datei bringen
reclock		Aktuellen Datensatz sperren / freigeben
ref		Referenten Funktion zur Bearbeitung von Sätzen (ref.rtf)
regi	X	Liste der logischen Register zur Auswahl anzeigen (adm.rtf)
rset	X	Der aktuellen Ergebnismenge einen anderen Namen geben (_start.flx)
rueck		Rückbuchung eines verliehenen Exemplars (ALFA) (d-wrtf.apr)
showbm		Zeige bookmark Datei (Hilfetexte – Lesezeichen)
signfile		Signalfile (.SGN) lesen und schreiben (org.rtf)
sortlist		Listenproduktion wie bei CockPit (exprtger.rtf)
summe	X	Summierung einer Kategorie über die ganze Ergebnismenge
switch		Umschalten zu einer anderen Datenbank (Schnellkopplung)

FLEX	*manuell startbar*	*Beschreibung*
Tabellenfunktionen		
tabedit		T-*.APR einlesen und zum Bearbeiten vorlegen
tabinit		Neue Tabellenstruktur einrichten
tabpick		Tabellenstruktur wählen
tabprod		Tabelle produzieren
tabstart		Anzeigefeld in eine Tabellen-Parameterdatei wandeln
tbllock		Schreibsperre für die gesamte Datenbank setzen
term		Terminfunktion aus kalender.flx
timeout		Ende der Sitzung, falls x Sekunden kein Eingabe (slice Prozedur)
uifedit	X	UIFe-Datei bearbeiten (adm.rtf)
utf8edit	X	Externes Editieren des Datensatzes mit Notepad. UTF-8 Format
uvar	X	*FLEX*-Datei lesen. Liste der Variablen anfertigen
vbfind	X	Volltextsuche in den Verlautbaungs-Dateien
Viewfunktionen: *vire.rtf*		
view0		Sichten der vorhandenen ViP-Dateien (Typ V-*.?PR)
view1		Erstellung eines View aus der aktuellen Ergebnismenge
view2		Erstellung eines View aus der aktuellen Ergebnismenge
viewexp		Aktuellen View exportieren
viewform		Verarbeitung eines ausgefüllten VIEW.FRM Formulars
viewold		Vorhandene View-Listen auswählen, sichten, löschen…
viewpara		Verarbeitung eines ausgefüllten VIEW.RTF Formulars
viewtab		Aktuellen View als Tabelle exportieren
weekday	X	Bestimmung des Wochentages eines beliebigen Datums von 1901 – 2099
winstart		Anwendung starten
xmlexp	X	Ergebnismenge oder aktuellen Satz als XML-Datei ausgeben

Tabelle 4: *FLEX*e