KERNFORSCHUNGSZENTRUM

KARLSRUHE

März 1975 KFK 2073

Institut für Datenverarbeitung in der Technik

Ein Verfahren zur modularen Kommandoentschlüsselung auf der Basis syntaxbeschreibender Tabellen

M. Rupp

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK 2073

Institut für Datenverarbeitung in der Technik

Ein Verfahren zur modularen Kommandoentschlüsselung auf der Basis syntaxbeschreibender Tabellen

M. Rupp

Gesellschaft für Kernforschung mbH, Karlsruhe

Kurzfassung

Der Aufwand für die Erstellung eines Kommandosystems kann wesentlich reduziert werden, wenn die Syntax der Kommandos in Tabellen abbildbar ist, die die Entschlüsselung eines Kommandowortes steuern.
Die Verwendung von Tabellen ermöglicht ferner eine funktionsunabhängige Neudefinition der Kommandosprache, sodaß jederzeit die Sprache an den Anlagen- und Systemausbau und an die Wünsche der Anwender angepaßt werden kann.
Der Bericht beschreibt die Struktur der syntaxbeschreibenden Tabellen, den Aufbau des Entschlüßlers und den Ablauf des Verfahrens.

Abstract

A Modular Command Decoding Method Based on Syntax Description Tables

The expense for the construction of command systems may be reduced noticeable by a table driven command decoding. The use of tables provides the ability of function independent redefinition of the command language, so that it may easily be adapted to a changing environment (installation and operating system) and to the needs of the users.
This report describes the structure of the syntax tables, the program moduls constituting the decoder and a demonstration of the working system.

Inhalt: Seite

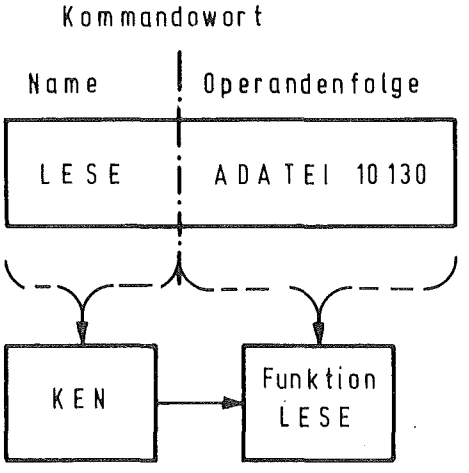
1. Einleitung

Bei kommandogesteuerten DV-Anlagen entfällt ein nicht unerheblicher Softwareaufwand auf die Entschlüsselung von Kommandos. Mit Hilfe von Kommandos steuert der Operator den Betrieb der Rechenanlage, der Benutzer verwendet Kommandos in der Steuersprache für den Batch-Betrieb und spricht über Kommandos die Funktionen eines interaktiven Systems an.

Bei der Vielzahl der notwendigen Entschlüsselungen von Kommandos innerhalb eines Systems liegt es nahe, einen modularen und möglichst universell einsetzbaren Kommandoentschlüssler zu erstellen. Wenn man von wenigen Ausnahmen (wie z.B. bei dem IBM-System TSO /1/) absieht, findet man bei den meisten Systemen eine funktionsspezifische Kommandoentschlüsselung. Geht man davon aus, daß ein Kommandowort aus einem Kommandonamen und einer Folge von Operanden besteht, so stellt Bild 1 den allgemein üblichen Ablauf der Entschlüsselung wie folgt dar:

Ein bestimmter System-Modul dekodiert den Kommandonamen und ruft die kommandoausführende Funktion auf, die individuell die Operanden entschlüsselt.

Mit dieser Organisationsform fällt bei der Erstellung eines Kommandosystems ein erheblicher Dekodierungsaufwand an, auch treten Schwierigkeiten bei der Einhaltung einheitlicher Kommandostrukturen auf (Normierungsvorschriften sind zu erstellen).



KEN: Kommandonamensentschlüsseler

BILD 1 Funktionsabhängige Entschlüsselung

Ferner ist eine nachträgliche Änderung der Kommandosyntax nur schwer möglich, da Eingriffe in den Ablaufcode unumgänglich sind.
Der vorliegende Bericht zeigt ein Verfahren zur modularen Entschlüsselung von Kommandos, bei dem die Entschlüsselung funktionslosgelöst und tabellengesteuert abläuft.
Das Verfahren wurde konzipiert für den Einsatz innerhalb eines Mehrfachzugriffssystems unter der Randbedingung, daß die Entschlüsselung weniger als 1K 16-Bit Worte an Speicherplatz benötigen darf.

2. Funktionsschema der Entschlüsselung

Im vorliegenden Verfahren erfolgt die Kommandoentschlüsselung in zwei Stufen durch die beiden Module:

- Kommandonamensentschlüßler
  und
- Operandenentschlüßler.

Die Module werden bei der Analyse des Kommandoworts gesteuert durch die zwei Tabellen:

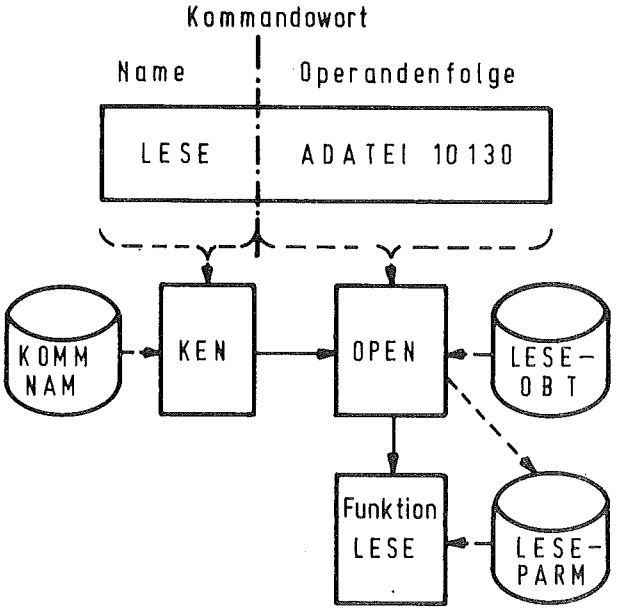
- Kommandonamensliste
- und
- Operandenbeschreibungstabelle.

Der Operandenentschlüssler legt die analysierten Operanden entsprechend der in der Operandenbeschreibungstabelle niedergelegten Vorschrift in dem
- Parameterblock
ab.

Der Parameterblock gehört zur kommandoausführenden Funktion, seine Struktur wird vom Funktionsersteller festgelegt und muß im Hinblick auf die Syntaxdefinition genau beschrieben sein.

Der Ablauf der Entschlüsselung ist aus Bild 2 ersichtlich.

Das Verfahren geht davon aus, daß das zu analysierende Kommandowort als Zeichenkette in einem systembekannten Puffer vorliegt.



KOMMNAM: Kommandonamensliste
OBT : Operandenbeschreibungstabelle
KEN : Kommandonamensentschlüsseler
OPEN : Operandenentschlüsseler

BILD 2 Funktionslosgelöste Entschlüsselung

Die Entschlüsselung beginnt damit, daß die Ablaufsteuerung des Kommandosystems den Kommandonamensentschlüßler aktiviert.
Der Kommandonamensentschlüßler findet den Kommandonamen über die Pufferanfangsadresse und führt einen Vergleich mit den in der Kommandonamensliste eingetragenen Namen durch.
In der Liste ist jeder Kommandoname mit einem Schlüssel versehen, der als systeminterner Name der kommandoausführenden Funktion anzusehen ist. Findet der Kommandonamensentschlüßler den Kommandonamen in der Liste, so gibt er den zugehörigen Schlüssel an die Ablaufsteuerung zurück.
Anhand des Schlüssels überprüft die Ablaufsteuerung die Zulässigkeit des Kommandos und weist es im Bedarfsfall zurück.

Ist das Kommando für die aktuelle Anwendung erlaubt, so ermittelt die Ablaufsteuerung die entsprechende Operandenbeschreibungstabelle und ruft den Operandenentschlüßler auf.
Die Aufgabe des Operandenentschlüßlers ist:

- Überprüfung der Operandenzeichenfolge auf syntaktische Richtigkeit,
- Entschlüsselung der erkannten Operanden
  und
- Ablage der ermittelten Werte in den funktionsspezifischen Parameterblock.

Der Ablauf der Entschlüsselung wird gesteuert durch die in der Operandenbeschreibungstabelle festgelegten Informationen über

- Syntax der Operandenfolge
  sowie
- Typ,
- Ablageplatz im Parameterblock und
- Standardwert der Operanden.

Nach erfolgreicher Entschlüsselung aktiviert die Ablaufsteuerung die angesprochene Funktion zur Ausführung des vorliegenden Kommandos.

Die beschriebene Organisationsform hat gegenüber der funktionsabhängigen Kommandoentschlüsselung folgende

Vorteile:

- Aufgrund der modularen Struktur der Kommandoentschlüsselung vermindert sich der Softwareaufwand bei der Erstellung eines Kommandosystems

und

- die Steuerung der Entschlüsselung durch funktionslosgelösten Tabellen ermöglicht eine rasche und leicht durchführbare Änderung der Kommandosyntax (auch während des Betriebes).

3. Struktur der Kommandosprache

In den eingangs skizzierten Systemen richtet sich die jeweilige Kommandoentschlüsselung nach dem individuellen Aufbau des Kommandowortes. Komplexe Kommandos erfordern einen hohen Dekodierungsaufwand, einfache Kommandos entsprechend weniger. Beim modularen Kommandoentschlüßler ergibt sich die umgekehrte Abhängigkeit, und zwar muß hier die Syntax des Kommandowortes nach den festliegenden Dekodierungsmöglichkeiten des Entschlüsselers ausgerichtet werden.

Damit ist die Möglichkeit der einsetzbaren Kommandosprache abhängig vom Ausbau bzw. Komfort des Kommandoentschlüsslers.

3.1 Zielsetzung

Die Konzeption des Verfahrens wurden darauf ausgerichtet, daß Kommandosprachen verwendet werden können, die folgende Anforderungen an den Entschlüßler stellen:

- Der Zeichenvorrat der Kommandosprache und die Bedeutung einzelner Zeichen (z.B. als Trennzeichen) wird bei Generierung festgelegt.
- Die Kommandos sind formatfrei angebbar.
- Die Bezeichnung festliegender Operanden wie auch die Kommandonamen sind frei wählbar und können abgekürzt angegeben werden.
- Operanden können bei eindeutiger Zuordnung stellungsfrei angegeben werden.

3.2 Aufbau des Kommandowortes

Der Einsatz des Verfahrens erfordert die Aufteilung des Zeichenvorrates A in die folgenden disjunkten Mengen:

Z = Menge der Ziffern = {0,1,2,3,4,5,6,7,8,9}
B = Menge der Buchstaben
T = Menge der Trennzeichen
L = {Leerzeichen} (einelementige Menge = {blank})
S = Menge der Sonderzeichen
O = Menge der Operandentrennzeichen
R = sonst
es gilt: $A = Z \cup B \cup T \cup L \cup S \cup O \cup R$

Wie bereits erwähnt, besteht ein Kommandowort aus dem Kommandonamen und evtl. aus einer Operandenfolge.
Die Operandenfolge setzt sich aus einem oder mehreren Operanden zusammen. Die Operanden können von unterschiedlichem Typ sein und zusätzlich mit sogenannten Bezeichnern versehen werden.

Ein Bezeichner wird i.a. dann einem Operanden vorangestellt, wenn eine eindeutige Zuordnung oder eine Hervorhebung notwendig ist. Ein Operand mit Bezeichnern heißt erweiterter Operand.

Als Regel für die Trennung der Operanden gilt, daß Operanden gegenüber dem Kommandonamen und untereinander durch ein oder mehrere Leerzeichen oder durch ein Trennzeichen - evtl. umgeben von Leerzeichen - zu trennen sind.

Der Kommandoname besteht aus einer trennzeichenfreien Zeichenfolge von maximal 15 Zeichen mit führendem Buchstaben.

An Operanden wird unter folgenden vier Typen unterschieden:

- ZAHL : stellt eine ganze, vorzeichenfreie Zahl dar, die noch in einem Maschinenwort darstellbar ist.

  Bei der Entschlüsselung wird der numerische Wert von ZAHL in den Parameterblock eingetragen.

  Bsp : 10,3,62345

- WORT : beinhaltet eine trennzeichenfreie, begrenzte Zeichenfolge mit führendem Buchstaben.

  Es wird die Gesamtheit der Zeichen in den dafür vorgesehenen Platz im Parameterblock eingetragen.

  Bsp : ADATEI , B3/15

- ZFOLGE : bezeichnet eine beliebige Zeichenfolge, die durch ein in der Zeichenfolge nicht verwendetes Sonderzeichen eingeschlossen wird. In den Parameterblock werden die beiden Pufferadressen des umschließenden Sonderzeichens eingetragen.

  Bsp : +GfK + , / 34+AB/

- KENNWORT : wird verwendet als Synonym eines kommandospezifisch festgelegten Wertes. Es wird gebildet wie ein WORT mit max. 15 Zeichen.
  Der vereinbarte Schlüssel (positive ganze Zahl) wird in den Parameterblock eingetragen.

  Bsp : NUM , ALLE , NEU1

Der Bezeichner eines Operanden ist aufgebaut wie das KENNWORT und ist damit ein Wortsymbol im Kommandowort. Zu jedem Bezeichner gehört ein Operandentrennzeichen, das unmittelbar zwischen dem Bezeichner und dem Operanden stehen muß.
Bsp: VON=op_i , SPALTE=op_j

Für die fest vereinbarten Elemente eines Kommandowortes, wie Kommandoname, KENNWORT und Bezeichner, können Abkürzungslängen festgelegt werden.
Die Abkürzungslänge besagt, wieviele Zeichen des Wortanfangs zur Identifizierung mindestens angegeben werden müssen.
Bsp:
gilt Wort=LESE mit Abkürzungslänge=2,
so ist LE,LES ubd LESE richtig, aber z.B.
L,LER und LESEN falsch.

Der exakte Aufbau eines zulässigen Kommandowortes ist in Backus-Naur-Form /2/ im Anhang dargestellt.

Entsprechend obiger Beschreibung gliedert sich beispielsweise das Kommandowort

SUCHE__ADATEI,AB=20_/_MEIER/_;_ALLE

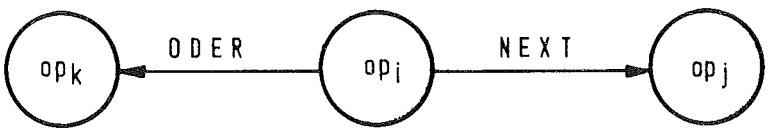
wie folgt:

SUCHE - Kommandoname
ADATAI - Wort
AB - Bezeichner
= - Operandentrennzeichen } erweiterter Operand } Operandenfolge
20 - ZAHL
/_MEIER/ - ZFOLGE
ALLE - KENNWORT
_ - Leerzeichen
,; - Trennzeichen
/ - Sonderzeichen

3.3 Anordnungen von Operanden

Die Beschreibung einer Kommandosyntax besteht neben der Typ-Angabe der Operanden in der Festlegung der Anordnungsfolgen der Operanden und der Kennzeichnung optionell angebbarer Operanden.

Welche Formen von Anordnungsfolgen vom Operandenentschlüßler verarbeitet und wie optionelle Operanden erkannt werden können, soll implizit am Vorgang der Entschlüsselung verdeutlicht werden.
Das Verfahren schreibt vor, daß jeder Operand mit den zwei Zusatzinformationen NEXT und ODER versehen wird. Diese Werte stellen die Kanten eines Syntaxgraphen dar, dessen Knoten die Operandentypen sind.



Die Bedeutung von ODER und NEXT wird ersichtlich aus dem Ablauf der Entschlüsselung:
Der Entschlüßler befindet sich quasi auf dem Knoten op_i und vergleicht die vorliegende Zeichenfolge mit dem Typ des Knoten op_i. Entspricht die Zeichenfolge dem Typ des Knoten, so gilt die Zeichenfolge als erkannt und der Entschlüßler geht über die Kante NEXT zum nächsten Knoten op_j. Im Knoten op_j setzt der Entschlüßler die Analyse der nächsten Zeichenfolge fort.
Wird im Knoten op_i keine Übereinstimmung festgestellt, so setzt der Entschlüßler die Analyse fort, wenn eine ODER-Kante von op_i wegführt. Er wandert zu dem mit ODER bezeichneten Knoten op_k und vergleicht die Zeichenfolge mit dem Typ von op_k.
Für den Fall, daß bei Ungleichheit keine ODER-Kante vom Knoten op_i wegführt, liegt eine ungültige Zeichenfolge vor, die nicht der vom Graphen beschriebenen Syntax genügt.

Der Syntaxgraph wird vervollständigt durch zusätzliche Steuerknoten, sogenannten Begrenzern, die z.B. Endpunkte der Entschlüsselung markieren. Auf die verschiedenen Funktionen der Begrenzer wird in 3.4 näher eingegangen.

Der Einsatz des Verfahrens bietet sich damit zur Entschlüsselung solcher Kommandosprache an, bei denen

- die auftretenden Operanden durch die vorliegenden Typen beschreibbar sind

und

- die zulässigen Anordnungsfolgen der Operanden durch einen Graphen obiger Art darstellbar sind.

3.4 Beispiele

Die folgenden Beispiele verdeutlichen die verschiedenen Einsatzmöglichkeiten der Begrenzer und sind gleichzeitig Anleitung zur Umsetzung einer Operandensyntax in einen entsprechenden Graphen.

Zur Beschreibung der Syntax wird folgende Notation verwendet:

op_i bezeichnen Operanden
Bezeichner stehen in Großbuchstaben

<...> in spitzen Klammern stehen optionelle Operanden
(...) in runden Klammern stehen stellungsfreie optionelle Operandenfolgen
[...] in eckigen Klammern stehen beliebig oft wiederholbare Operanden bzw. Operandenfolgen
{...} in geschweiften Klammern stehen alternative Operanden bzw. Operandenfolgen

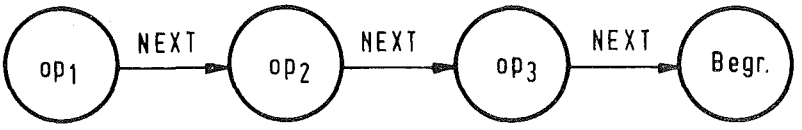
Für die Beispiele werden folgende Operanden gewählt:

op_1, op_2 : Typ : ZAHL
op_3 : Typ : KENNWORT Werte:ALLE,ERSTER
op_4, op_5 : Typ : ZFOLGE
op_6 : Typ : WORT
*) Es gilt folgende Zeicheneinteilung:
Trennzeichen sind ',' und ';'
Sonderzeichen sind '/', '*' und '+'
Operandentrennzeichen sind ':' und '='

Bsp 1:

Operandensyntax: op_1 op_2 op_3
richtige Folge : 10,20,ALLE
falsche Folge : 33 ERSTER

Syntaxgraph:

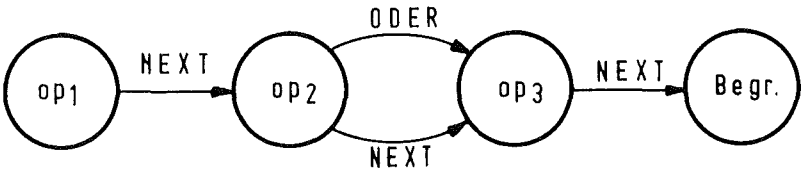


Anm.: Wird ein Begrenzer ohne Kanten erreicht, so ist die Entschlüsselung dann erfolgreich verlaufen, wenn keine Zeichen mehr zu analysieren sind - ansonsten liegt ein Fehler vor.

Bsp 2:

Operandensyntax: op_1 <op_2> op_3
richtige Folge : 22;ERSTER

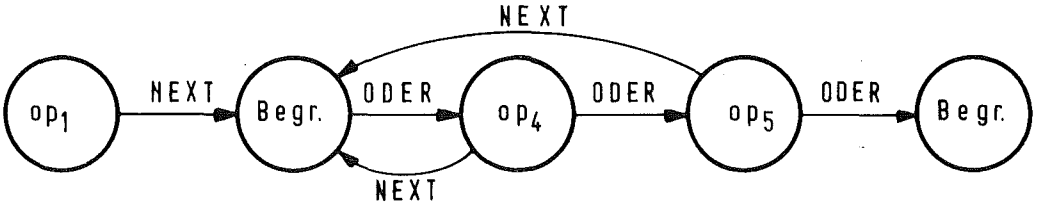
Syntaxgraph:



Bsp 3:

Operandensyntax: op_1 (VON:op_4 BIS:op_5)
richtige Folgen: 10 VON:/EMMA/ BIS:+ANNA+
30;BIS:++ VON:/3/
135 BIS:*10/30*
352

Syntaxgraph:



Anm: Die Begrenzer-Kante ODER besagt, daß die Entschlüsselung beim durch ODER bezeichneten Knoten fortgesetzt werden soll, sofern noch Zeichen zu analysieren sind.

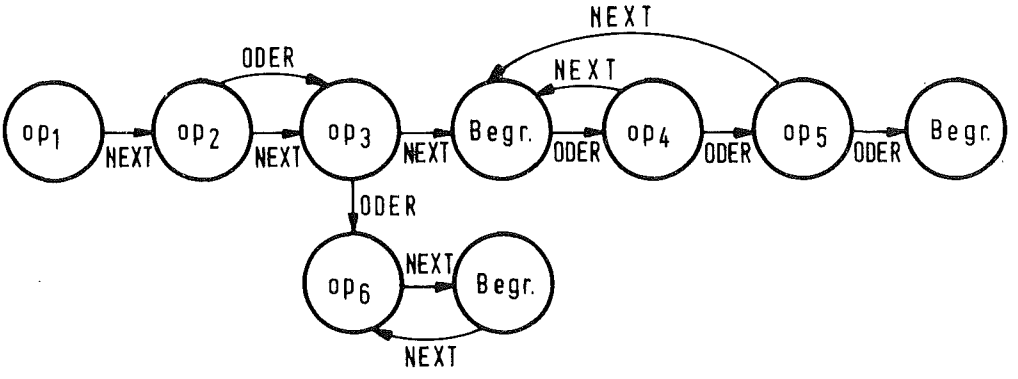
Bsp 4:

Operandensyntax:

$$op_1 \quad \langle op_2 \rangle \quad \left\{ \begin{array}{l} op_3 \left(\left\{ \begin{array}{l} \text{VON:} \\ \text{FROM=} \end{array} \right\} op_4 \left\{ \begin{array}{l} \text{BIS:} \\ \text{TO} \end{array} \right\} op_5 \right) \\ \\ [op_6] \end{array} \right\}$$

richtige Folgen: 10 ALLE BIS:/Z1/ FROM=*A B*
20;30,ADATEI
10 BF ATE;G1V

Syntaxgraph:



Anm: Die Begrenzer-Kante NEXT bewirkt bei nicht vollständig entschlüsseltem Zeichenpuffer die Unterbrechung der Entschlüsselung. Die angesprochene Funktion wird aktiviert und verarbeitet den Operanden op_6. Danach wird die Entschlüsselung der restlichen Zeichenfolge bei dem durch NEXT bezeichneten Knoten, also wieder bei op_6 fortgesetzt. Diese Abarbeitungsregel erlaubt somit die Verwendung beliebig oft wiederholbarer Operanden (-folgen),die nacheinander abgearbeitet werden.

4. Realisierung

4.1 Kommandonamensentschlüsselung

Die Aufgabe des Kommandonamensentschlüßlers besteht in der Hauptsache im Durchsuchen einer Namensliste nach dem Suchobjekt 'Kommandoname', bzw. in der Ermittlung des kommandobeschreibenden Schlüssels.

4.1.1 Struktur der Namenstabellen

Da die Suche nach einem Wortsymbol auch bei der Auflösung eines Operanden vom Typ KENNWORT und bei der Erkennung eines Bezeichners anfällt, wurde die Struktur der Namenstabellen, wie

- die globale Kommandonamensliste

und

- die kommandospezifischen KENNWORT- und Bezeichnertabellen,

gleichartig gestaltet.

Die Namenstabellen bestehen aus aneinandergereihten Namenselementen, deren Aufbau im Bild 3 dargestellt ist.

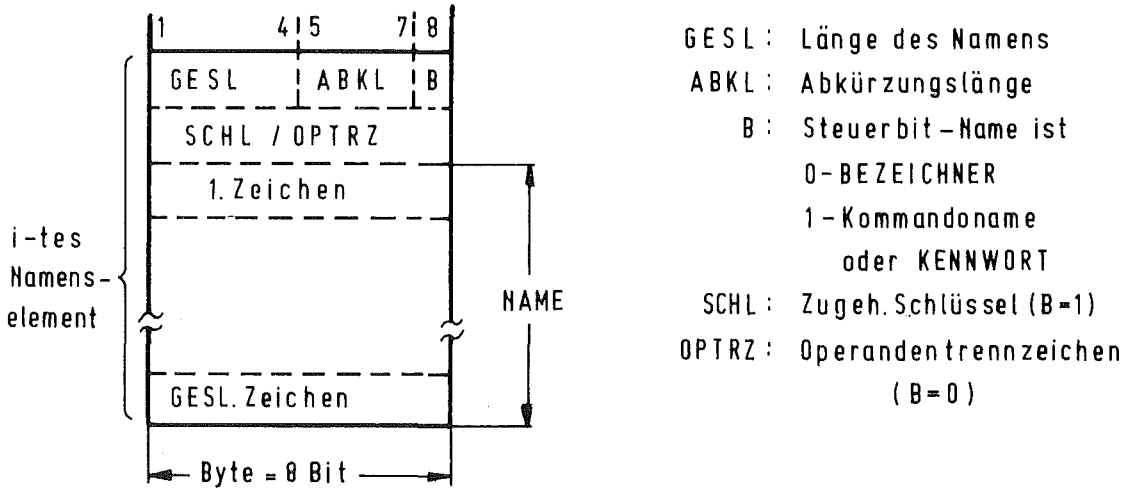


BILD 3 Darstellung eines Namens in einer Namensliste

Das Steuerbit B gibt an, ob Byte 2 einen Schlüssel oder ein Operandentrennzeichen enthält. GESL bezeichnet die Gesamtlänge, ABKL die Minimallänge des Namens.
Übereinstimmung zwischen Suchobjekt S ($S=s_1\ s_2 \ldots s_m$) und Namen ($N=n_1\ n_2 \ldots n_{GESL}$) liegt dann vor, wenn gilt:

$ABKL \leq m \leq GESL$ und
$s_i = n_i$ für $i = 1(1)ABKL$
$s_j = n_j$ für $j = ABKL+1(1)m$

Bild 3 basiert auf einer Implementierung, bei der ein Zeichen in einem Byte (8-Bit) darstellbar ist.
Für die Beschreibungsgrößen ergibt sich folgender Wertebereich:

$0 \leq ABKL < 8$, ABKL=0 bedeutet: ABKL=GESL
$0 \leq GESL < 16$
$0 \leq SCHL < 256$.

Es ist zu beachten, daß jedes Namenselement an einer Wortgrenze beginnt und daß die Tabelle mit dem GESL-Wert 0 abschließt.

4.1.2 Suchvorgang

Die Überprüfung der Suchobjekte Kommandoname, KENNWORT oder Bezeichner auf Zulässigkeit wird vom Modul FIND anhand der entsprechenden Tabelle durchgeführt.
Dem Modul werden bei Aufruf die Adressen des Suchobjekts und der Namenstabelle übergeben. Gesteuert durch die Größen GESL und ABKL führt er in der Tabelle eine sequentielle Suche durch. FIND beendet den Suchvorgang, sobald Name und Suchobjekt gemäß obigen Regeln übereinstimmen.
FIND gibt bei positivem Vergleichsergebnis im Falle eines Kommandonamens bzw. KENNWORT'es den Schlüssel, im Falle eines Bezeichners - vorausgesetzt, das Operandentrennzeichen stimmt überein - einen Returncode > 0 an das aufrufende Programm zurück.
Erreicht FIND das Ende der Tabelle (GESL=0), so gibt er den Returncode -1 zurück als Zeichen dafür, daß keine Übereinstimmung festgestellt werden konnte.

4.1.3 Ablauf

Der Vorgang der Kommandonamensentschlüsselung besteht somit

- im Überlesen von führenden Leerzeichen im Kommandopuffer
und
- im Aufruf des Moduls FIND mit Versorgung der Anfangsadressen der Kommandonamensliste und des ersten Zeichens im Kommandopuffer.

4.2 Operandenentschlüsselung

Die Vorgehensweise des Operandenentschlüßlers wird gesteuert durch die in der Operandenbeschreibungstabelle niedergelegten Vorschriften über Syntax und Bedeutung der Operanden.

4.2.1 Aufbau der Operandenbeschreibungstabelle

Jedes Kommando, bei dem Operanden angebbar sind, benötigt eine Operandenbeschreibungstabelle (OBT).
Die Operandenbeschreibungstabelle enthält die Darstellung der Operandensyntax und die Bezüge der Operanden zum Parameterblock (s. Bild 4).

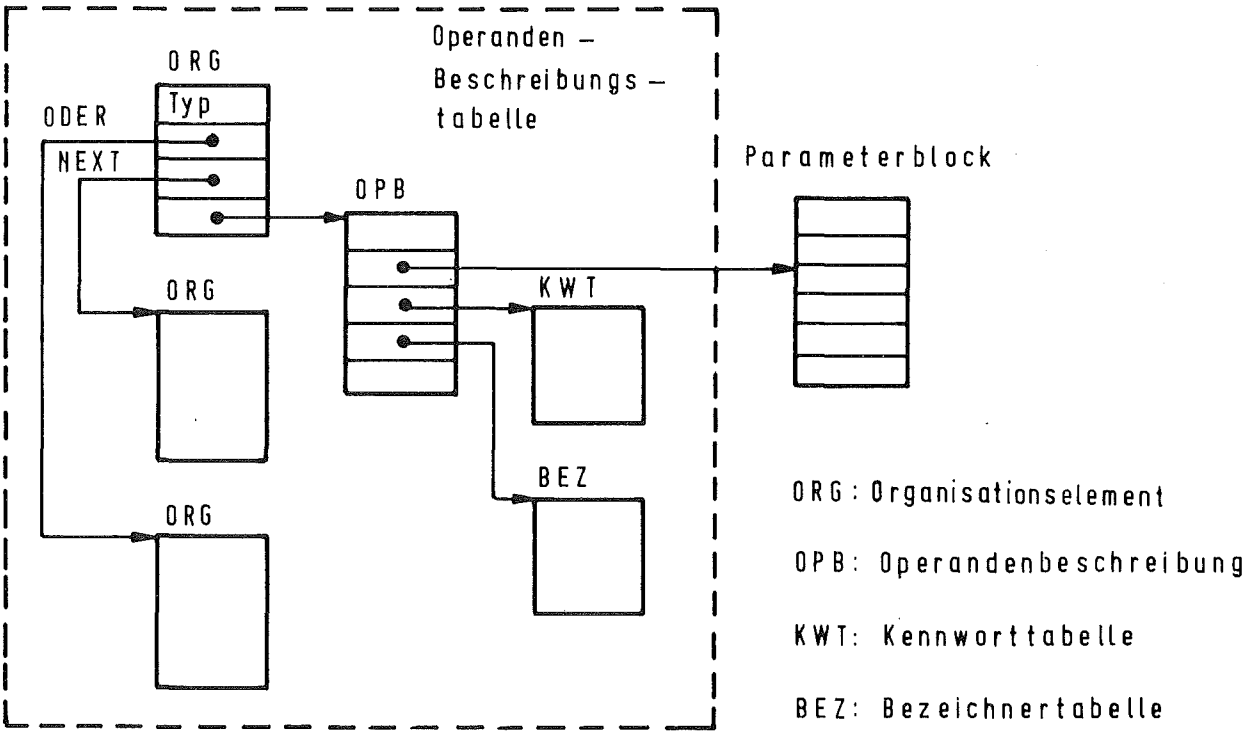


BILD 4 Struktureller Aufbau der Operandenbeschreibungstabelle

Die Operandenbeschreibungstabelle besteht aus sogenannten

- Organisationselementen,
- Operandenbeschreibungen

und gegebenenfalls aus

- KENNWORT- und Bezeichnertabellen.

Organisationselement

Die Organisationselemente beschreiben den der Operandensyntax zugrunde liegenden Graphen. Ein Organisationselement beinhaltet den Typ des Knoten, die Kanten ODER und NEXT, sowie den Verweis zu der Operandenbeschreibung des darzustellenden Operanden (INF).

Es gibt sechs Typen von Organisationselementen, die wie folgt unterschieden werden:

Typ 1 beschreibt einen Operanden vom Typ ZAHL
Typ 2 beschreibt einen Operanden vom Typ WORT
Typ 3 beschreibt einen Operanden vom Typ ZFOLGE
Typ 4 beschreibt einen Operanden vom Typ KENNWORT
Typ 5 beschreibt einen erweiterten Operanden
Typ 6 beschreibt einen Begrenzer

Ein Organisationselement wird in PL/1-Schreibweise dargestellt durch folgende Struktur:

```
DCL 1 ORGEL,              /*Organisationselement*/
      2 TYP  bin fixed,   /*1<=TYP≤6*/
      2 ODER bin fixed,   /*Verweis auf alternatives Element*/
      2 NEXT bin fixed,   /*Verweis auf nächstes Element*/
      2 INF  bin fixed;   /*Verweis auf zugeh. Operandenbeschreibung*/
```

Die Verweise ODER,NEXT und INF sind Distanzadressen bezogen auf den Anfang der Operandenbeschreibungstabelle. Entfällt ein Verweis, so muß der entsprechende Wert < 0 gesetzt werden.

Operandenbeschreibung

Die Operandenbeschreibung stellt die Schnittstelle zwischen Entschlüsselung und Funktion dar. Sie ist typabhängig strukturiert und enthält in allen Fällen den Standardwert des Operanden sowie die Adresse des Parameterblocks, ab der der entschlüsselte Wert abzulegen ist.
Die vier unterschiedlichen Operandentypen und der erweiterte Operand lassen sich durch folgende PL/1-Strukturen beschreiben:

1) Typ 1 - ZAHL
   Struktur:

```
DCL 1 OPBZAHL, /*Operandenbeschreibung ZAHL*/
      2 STW  bin fixed
      2 POSZ bin fixed;
```

   mit
   - STW : Standardwert des Operanden
   - POSZ: Ablageadresse des numerischen Wertes von ZAHL im Parameterblock (Distanzadresse bezüglich des Parameterblockanfangs).

2) Typ 2 - WORT
   Struktur:

```
DCL 1 OPBWORT, /*Operandenbeschreibung WORT*/
      2 STW  bin fixed,
      2 POSW bin fixed,
      2 MAXL bin fixed;
```

   mit
   STW - Standardwert des Operanden
   POSW - Parameterblockadresse (Distanz) ab der WORT eingetragen wird
   MAXL - maximal zulässige Zeichenlänge von WORT (damit Länge des entspr. Parameterablageplatzes).

3) Typ 3 - ZFOLGE
   Struktur:

```
DCL 1 OPBZFOLGE, /*Operandenbeschreibung ZFOLGE*/
      2 STW  bin fixed,
      2 POSA bin fixed,
      2 POSB bin fixed;
```

mit

STW - Standardwert des Operanden

POSA - Parameterblockadresse (Distanz), ab der die Pufferadresse des führenden Sonderzeichens von ZFOLGE abgelegt wird.

POSB - Parameterblockadresse (Distanz), ab der die Pufferadresse des abschließenden Sonderzeichens von ZFOLGE abgelegt wird.

4) Typ 4 - KENNWORT

Struktur:

```
DCL 1 OPBKENNWORT, /*Operandenbeschreibung KENNWORT*/
      2 STW   bin fixed,
      2 POSK  bin fixed,
      2 KWTAB bin fixed;
```

mit

STW - Standardwert des Operanden

POSK - Ablageadresse (Distanz), des KENNWORT-Schlüssels im Parameterblock

KWTAB- Bezug zur Tabelle, die die KENNWORT'e des Operanden enthält (Distanzadresse bezogen auf den OBT-Anfang)

5) Typ 6 - erweiterter Operand

Struktur:

```
DCL 1 OPERW, /*Operandenbeschreibung eines erweiterten Operanden*/
      2 STW    bin fixed,
      2 POS1   bin fixed,
      2 POS2   bin fixed,
      2 BEZTAB bin fixed,
      2 OPTYP  bin fixed;
```

mit

STW - Standardwert des erweiterten Operanden

BEZTAB - Bezug zur Tabelle, die die Bezeichner des erweiterten Operanden enthält (Distanzadresse bezogen auf den Anfang der OBT).

OPTYP - Typ des durch den Bezeichner gekennzeichneten Operanden

POS1 - entspricht bei OPTYP=1:POSZ
entspricht bei OPTYP=2:POSW
entspricht bei OPTYP=3:POSA
entspricht bei OPTYP=4:POSK
POS2 - bedeutungslos bei OPTYP=1
entspricht bei OPTYP=2:MAXL
entspricht bei OPTYP=3:POSB
entspricht bei OPTYP=4:KWTAB

4.2.2 Ablauf

Der Operandenentschlüßler muß vom aufrufenden Programm mit folgenden Adressparametern versorgt werden:

- Adresse der Operandenbeschreibungstabelle,
- Adresse des Organisationselements, bei dem die Analyse aufgenommen wird (AUFSETZP),
- Adresse des Parameterblocks,
- Adresse des Puffers, dessen Inhalt entschlüsselt werden soll (PUFFERP).

Der Operandenentschlüßler endet mit der Rückgabe eines ablaufspezifischen Returncodes an das aufrufende Programm.

Die Tätigkeit des Operandenentschlüßlers besteht im wesentlichen im Vergleichen

- der aktuellen Zeichenfolge, d.h. der noch nicht entschlüsselten Zeichenfolge bis zum nächsten Trennzeichen,

mit

- dem Typ des aktuellen Organisationselements.

PUFFERP adressiert den Anfang der ersten aktuellen Zeichenfolge, AUFSETZP bestimmt das erste aktuelle Organisationselement.
Wird beim Vergleich Übereinstimmung von Zeichenfolge und Typ festgestellt, so wird der Wert der aktuellen Zeichenfolge (d.h. Zahl, WORT, Adresse oder Schlüssel) in den laut Operandenbeschreibung vorgeschriebenen Platz im Parameterblock eingetragen. Danach werden das durch NEXT gekennzeichnete Organisationselement und die folgende Zeichenfolge aktuell und ein neuer Vergleich wird angestellt.

Entspricht die aktuelle Zeichenfolge nicht dem Typ des aktuellen Organisationselements, so wird bei

- ODER < 0
  die Analyse abgebrochen. In PUFFERP wird die Anfangsadresse der unzulässigen Zeichenfolge geschrieben und als Returncode wird die Parameterblockadresse zurückgegeben, ab der ein gültiger Wert eingetragen worden wäre. Ferner enthält AUFSETZP den NEXT-Verweis des letzen Organisationselements, bei dem Übereinstimmung festgestellt wurde.
  Aufgrund dieser Informationen kann das aufrufende Programm eine Aufforderung zur Verbesserung der fehlerhaften Zeichenfolge anstoßen und danach die Wiederaufnahme der Dekodierung der verbesserten Zeichenfolge veranlassen.

- ODER > = 0
  das durch ODER gekennzeichnete Organisationselement aktuell, d.h. die Analyse wird mit dem Vergleich der Zeichenfolge mit einem alternativen Element fortgesetzt.

Wird ein Organisationselement vom Typ Begrenzer aktuell, so werden zwei Fälle unterschieden:

Fall 1 - sind alle Zeichen im Puffer analysiert - die aktuelle Zeichenfolge ist leer -, so gilt unabhängig der Werte von NEXT und ODER die Entschlüsselung als erfolgreich beendet. Der Entschlüßler gibt den Returncode-5 zurück.

Fall 2 - besteht die aktuelle Zeichenfolge noch aus Zeichen, die ungleich dem Leerzeichen sind, so wird bei

- NEXT >=0: die Entschlüsselung mit dem Returncode-4 unterbrochen. AUFSETZP erhält die Adresse des mit NEXT bezeichneten Organisationselements, PUFFERP die Adresse der aktuellen Zeichenfolge.
- NEXT <0 und ODER <0: die Entschlüsselung mit dem Returncode-3 abgebrochen.
- NEXT <0 und ODER >0: das durch ODER bezeichnete Organisationselement aktuell und die Analyse fortgesetzt.

Der Operandenentschlüßler bricht auch dann die Analyse ab, wenn

- die aktuelle Zeichenfolge leer ist und das aktuelle Organisationselement kein Begrenzer ist und der ODER-Wert < 0 ist (Returncode = -2) und wenn
- Trennzeichen unmittelbar aufeinander folgen (Returncode = -1).

Von der oben beschriebenen Vorwärtsanalyse wird in zwei Fällen abgewichen:

1) Wird ein Organisationselement vom Typ 1-3 (ZAHL, WORT, ZFOLGE) mit dem ODER-Wert < 0 erreicht und wird keine Übereinstimmung mit der aktuellen Zeichenfolge festgestellt, so wird untersucht, ob der zuletzt analysierte Operand vom gleichen Typ ist und einen ODER-Wert > = 0 besitzt. Ist dies der Fall, so wird diesem Operanden sein Standardwert zugewiesen, während dem aktuellen Operanden der bereits entschlüsselte Wert zugeordnet wird. Die Entschlüsselung, die ansonsten abgebrochen worden wäre, wird bei dem durch NEXT bezeichneten Organisationselement fortgesetzt.
   Damit wird die Abbildung folgender Operandensytanx ermöglicht:

   $$< op_1 > \; op_2$$

   mit op_1 und op_2 vom gleichen Typ.

2) Da die Operandentypen WORT und KENNWORT vom Aufbau her nicht unterscheidbar sind, läuft die Entschlüsselung gemäß folgender Vorschrift ab:
   Die aktuelle Zeichenfolge wird nur dann dem aktuellen Organisationselement vom Typ 2 (WORT) zugeordnet, wenn auf keinem vom Element ausgehenden ODER-Pfad ein Element vom Typ 4 (KENNWORT) erreicht werden kann, das als KENNWORT die Zeichenfolge verwendet.
   Damit ist das KENNWORT vor dem WORT priorisiert.

5. Beispiele zur Anwendung des Verfahrens

Anhand der folgenden Beispiele sollen sowohl Aufbau und Struktur der syntaxbeschreibenden Tabellen als auch der Ablauf der Entschlüsselung näher verdeutlicht werden.

5.1 Kommandonamensentschlüsselung

Die beiden Funktionen F_1 und F_2 eines Kommandosystems sollen mit folgenden Namen aufrufbar sein (Schlüsselzuordnung: F_1-1, F_2-2):

Name	Schlüssel	Abkürzungslänge
STOP	1	4
START	2	0
AUS	1	1
AN	2	1

STOP und START haben demnach die Alias-Namen AUS bzw. AN.

Da bei n Namen n! unterschiedliche Anordnungen der Namenselemente in der Kommandonamensliste möglich sind, sind hier 24 Lösungen angebbar.

Bei den beiden folgenden Lösungen wird vorausgesetzt, daß eine 16-Bit-Wort Adressierung vorliegt, wobei ein Zeichen in einem Byte (8 Bit) darstellbar ist. Der Aufbau der Namenselemente entspricht der in Bild 3 gegebenen Darstellung.

Lösung 1

Adresse	Inhalt
Ø	4/4/1
	1
	ST
	OP
4	5/Ø/1
	2
	ST
	AR
	T
9	3/1/1
	1
	AU
	S
13	2/1/1
	2
	AN
16	Ø

Lösung 2

Adresse	Inhalt
Ø	2/1/1
	2
	AN
3	5/Ø/1
	2
	ST
	AR
	T
8	4/4/1
	1
	ST
	OP
12	3/1/1
	1
	AU
	S
16	Ø

Aufgrund der sequentiellen Suchmethode des Moduls FIND wird in

Lösung 1: A als AUS
Lösung 2: A als AN

erkannt.

Der effektive Einsatz des Verfahrens erfordert hinsichtlich der Suchstrategie eine gezielte Anordnung der Namen.

Dies gilt insbesondere im Hinblick auf die Minimierung der Suchzeiten für häufig benutzte Kommandonamen.

5.2 Operandenentschlüsselung

Es wird gezeigt, wie die in 3.4 Bsp. 4 gewählte Operandensyntax in einer Operandenbeschreibungstabelle dargestellt werden kann.

Das Kommando mit der Operandensyntax

$$op_1\ \langle op_2\rangle\ \left\{\begin{array}{l} op_3\ \left(\left\{\begin{array}{l}\text{VON:}\\\text{FROM=}\end{array}\right\}\ op_4\ \left\{\begin{array}{l}\text{BIS:}\\\text{TO=}\end{array}\right\}\ op_5\right) \\ \\ [op_6] \end{array}\right\}$$

dient zur Aktivierung einer Funktion, bei der unterstellt wird, daß eine Beschreibung mit folgender alternativer Parameterstruktur vorliegt:

Fall 1

Distanzadr. Parameterbl.	Typ	Standard-Wert	Bedeutung
Ø	ZAHL	-	Anfangsnummer
1	ZAHL	Ø	Endnummer
2	WORT	-	wiederholbarer Dateiname mit max. 6 Zeichen

Fall 2

Distanzadr. Parameterbl.	Typ	Standard-Wert	Bedeutung
Ø	ZAHL	-	Anfangsnummern
1	ZAHL	Ø	Endnummer
2	KENNUNG	1	1- erstes Element Ø- alle Elemente
3	Anf.-Adr.	-1	Stringanfang
4	End.-Adr. ZFOLGE		
5	Anf.-Adr.	-1	Stringende
6	End.-Adr. ZFOLGE		

Die folgende Darstellung zeigt den Aufbau einer Operandenbeschreibungstabelle, aufgrund derer der Operandenentschlüßler eine Operandenfolge gemäß obigen Vorschriften entschlüsselt.

Die Adressbezüge sind in der Darstellung durch die symbolische Namen der angesprochenen Elemente ersetzt, NULL bedeutet das Fehlen der Adresse bzw. stellt einen negativen Adreßbezug (s. 4.2.1) dar.
Für die Darstellung der Organisationselemente wurde gemäß 4.2.1 folgende Form gewählt

```
ORG_nr      (symbolischer Name eines Organisationselements)
   TYP
   ODER
   NEXT
   INF
```

Die Operandenbeschreibung wird im komplexesten Fall dargestellt durch:

```
OPB_nr      (symbolischer Name einer Operandenbeschreibung)
   STW
   POS1
   POS2
   BEZTAB
   OPTYP ,
```

ansonsten typabhängig gemäß 4.2.1 reduziert.

Die KENNWORT- und Bezeichnertabellen haben den symbolischen Namen KWT_nr bzw. BEZ_nr und sind aufgebaut wie die Kommandonamenslisten (s. 5.1).

Die resultierende Operandenbeschreibungstabelle ist:

```
ORG_1
   1
   NULL
   ORG_2
   OPB_1

ORG_2
   1
   ORG_3
   ORG_3
   OPB_2

ORG_3
   4
   ORG_8
   ORG_4
   OPB_3

ORG_4
   6
   ORG_5
   NULL
   NULL

ORG_5
   5
   ORG_6
   ORG_4
   OPB_5

ORG_6
   5
   ORG_7
   ORG_4
   OPB_6

ORG_7
   6
   NULL
   NULL
   NULL
```

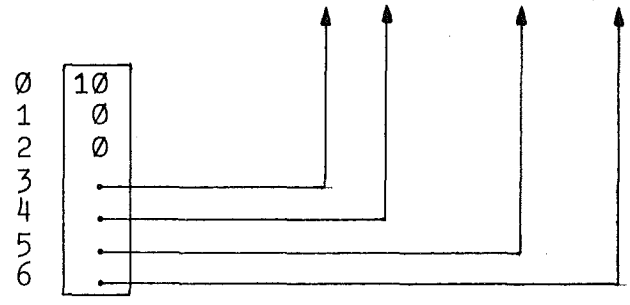
```
OPB_1
   -1
   Ø

OPB_2
   Ø
   1

OPB_3
   1
   2
   KWT_3

OPB_5
   -1
   3
   4
   BEZ_5
   3

OPB_6
   -1
   5
   6
   BEZ_6
   3
```

```
KWT_3
   4/1/1
   Ø
   AL
   LE
   6/1/1
   1
   ER
   ST
   ER
   Ø

BEZ_5
   3/1/Ø
   :
   VO
   N
   4/2/Ø
   =
   FR
   OM
   Ø

BEZ_6
   3/1/Ø
   :
   BI
   S
   2/2/Ø
   =
   TO
   Ø
```

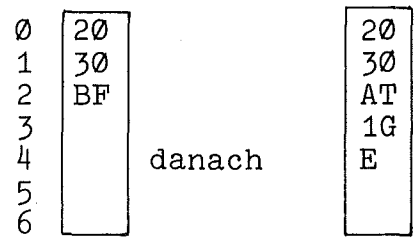
```
ORG_8                    OPB_8
   2                        -1
   NULL                     2
   ORG_9                    6
   OPB_8

ORG_9
   6
   NULL
   ORG_8
   NULL
```

Unter der Voraussetzung, daß ORG_1=AUFSETZP gilt, erzeugt der Operandenentschlüßler beispielsweise folgende Parameterblock-einträge:

Operandenfolge: 1Ø ALLE BIS: /Z1/ FROM=*A 13*



Operandenfolge: 2Ø 3Ø BF AT1GE



6. Zusammenfassung

Vom Anwender des Verfahrens wird verlangt, daß er sich genaue Kenntnisse über die Abarbeitungsregeln von Kommandonamens- und Operandenentschlüßler sowie über Form und Aufbau der syntaxbeschreibenden Tabellen verschafft.
Als Hilfsmittel zur Erstellung der Tabellen steht ein Programmsystem /3/ zur Verfügung, das im Institut für Datenverarbeitung in der Technik im Kernforschungszentrum Karlsruhe entwickelt wurde. Das System gestattet auf einfache Art und Weise die Generierung der sprachbeschreibenden Tabellen (Kommandonamenslisten und Operandenbeschreibungstabellen) und ermöglicht das Austesten der definierten Kommandosprache (Verifikation der Syntaxabbildung in die Tabellen).

Die vorangegangenen Beispiele haben gezeigt, daß das Verfahren die Darstellung und Entschlüsselung selbst komplexer Kommandosprachen erlaubt.
Aufgrund der relativ einfachen Struktur der syntaxbeschreibenden Tabellen benötigt der Entschlüßler weniger als 1 K 16-Bit-Worte an Speicherplatz und ist somit besonders für den Einsatz auf kleineren Rechenanlagen geeignet.

Das beschriebene Verfahren ist der funktionsabhängigen Kommandoentschlüsselung in vielfacher Hinsicht überlegen.
Die klaren Schnittstellen zwischen Kommandoentschlüsselung und kommandoausführender Funktion erleichtern die Konzeption und die Erstellung eines Kommandosystems und gewährleisten eine problemlose Integration nachträglich geschriebener Funktionen in das Kommandosystem.
Mit der Möglichkeit der Neugenerierung der syntaxbeschreibenden Tabellen kann das Kommandosystem den vielfältigsten Anforderungen angepaßt werden, ohne daß Eingriffe in den Code der kommandoausführenden Funktionen notwendig sind.
Im einzelnen bedeutet dies, daß der Systeminstallateur das Kommandosystem z.B. durch Sperren von Parametern an den augenblicklichen Anlagen- bzw. Systemausbau anpassen kann. Ferner kann er verschieden

mächtige Kommandosprachen generieren, die unterschiedlich privilegierten Benutzerklassen die mehr oder weniger eingeschränkte Benutzung der gleichen Funktionen ermöglichen.
Dem Anwender bringt das Verfahren den Vorteil, daß die Kommandosprache seinen Wünschen und seinem Sprachverständnis entsprechend definiert werden kann.

Zusammenfassend läßt sich sagen, daß der Einsatz des modularen Kommandoentschlüßlers in einem Kommandosystem aufgrund der Verwendung von modifizierbaren syntaxbeschreibenden Tabellen dazu beiträgt, daß das Kommandosystem

- einfacher erstellbar,
- schnell erlernbar,
- leicht handhabbar,

und

- effektiv einsetzbar ist.

Literatur

/1/ Guide to Writing a Terminal Monitor Program or a Command Processor
IBM Systems Reference Library

/2/ J.W. Backus, u.a.
Revised Report on the Algorithmic Language ALGOL 60
Communications of the ACM 6 (1963), 1

/3/ M. Rupp
Programmbeschreibung eines tabellengesteuerten Kommandoentschlüßlers
Externer Bericht 13/74-2, Dezember 1974

Anhang

Darstellung eines Kommandowortes in BNF

Anmerkung zur Notation:

- Kleinbuchstaben stellen Elemente des in 3.2 beschriebenen Zeichenvorrats dar, z.B. bedeutet t ein Element aus T,
- durch '|' werden Alternativen getrennt,
- 'ε' ist das Leerterminal.

Es gilt:

```
<Kommandowort>      ::= <TR1><Kommandoname><TR1>
                        <Operandenfolge><TR1>
<Kommandoname>      ::= <KENNWORT>
<TR1>               ::= l|l<TR1>|ε
<Operandenfolge>    ::= <TR2><OP>|<TR2><OP>
                        <Operandenfolge>|ε
<TR2>               ::= t|l|<TR1>t<TR1>
<OP>                ::= <OPN>|<OPERW>
<OPN>               ::= <ZAHL>|<WORT>|<ZFOLGE>|
                        <KENNWORT>
<OPERW>             ::= <BEZEICHNER>σ<OPN>
<ZAHL>              ::= z|z<ZAHL>
<WORT>              ::= b|b<W1>
<W1>                ::= <W2><W1>
<W2>                ::= b|s|z|r|ε
<ZFOLGE>            ::= <S><ZF1><S>
<ZF1>               ::= a|a<ZF1>|ε      mit a≠S
<S>                 ::= s
<KENNWORT>          ::= <WORT>          mit Länge (WORT)<15
<BEZEICHNER>        ::= <KENNWORT>
```