

DIN DEUTSCHES INSTITUT FÜR NORMUNG E.V.

DIN 66 265

Informationsverarbeitung

Schnittstellen eines Kerns für transaktionsorientierte Anwendungssysteme (KDCS-TAS-Kern)

Normenausschuß Informationsverarbeitungssysteme (NI)
im DIN Deutsches Institut für Normung e.V.

BEUTH VERLAG GMBH · BERLIN

Informationsverarbeitung
**Schnittstellen eines Kerns für transaktionsorien-
 tierte Anwendungssysteme (KDCS-TAS-Kern)**

DIN
66 265

Information processing; interface for a transaction-oriented system-kernel

Inhalt

	Seite		Seite
0 Anwendungsbereich und Zweck	2	2.4 Freilaufende Vorgänge	
0.1 Einleitung	2	und die FPL-Schnittstelle	43
0.2 Externe Charakterisierung eines		3 Formale Beschreibung des KDCS-TAS-Kerns . . .	47
transaktionsorientierten Anwendungssystems . .	2	Inhaltsverzeichnis	47
0.3 Aufgaben eines TAS-Kerns	5	3.1 Einführung	48
0.4 Der Begriff der TAS-Definition	7	3.2 Definition der Beschreibungsmittel	49
1 Allgemeines zur Beschreibung, Bezeichnung . . .	8	3.3 Gemeinsame Bestandteile des Netzmodells . . .	64
1.1 Allgemeines zur Beschreibung	8	3.4 Teilsysteme (Prädikate und Transitionen)	80
1.2 Bezeichnung	8	3.5 Die normrelevanten Anteile	
2 Verbale Beschreibung des KDCS-TAS-Kerns . . .	9	des formalen Systemmodells	125
Inhaltsverzeichnis	9	4 Tabellen	127
2.1 Besonderheiten des KDCS-TAS-Kerns,		5 Spracheinbettung COBOL	134
Grundlegendes zu den		Zitierte Normen und andere Unterlagen	146
Programmlauf-Schnittstellen	11	Weitere Normen	146
2.2 Die bei einer Systemdefinition einzubringenden		Erläuterungen	146
Materialien und festzulegenden Parameter . . .	16	Stichwortverzeichnis	147
2.3 Dialogvorgänge und die DPL-Schnittstelle	19		

Internationale Patentklassifikation

G 06 F 3/00
 G 06 F 12/16

Fortsetzung Seite 2 bis 152

Normenausschuß Informationsverarbeitungssysteme (NI) im DIN Deutsches Institut für Normung e.V.

Jede Art der Vervielfältigung, auch auszugsweise, nur mit Genehmigung des DIN Deutsches Institut für Normung e.V., Berlin, gestattet.

läuterungen. Typische Beispiele für transaktionsorientierte Anwendungssysteme sind Buchungssysteme und Anwendungssysteme im Bankwesen mit Dialogbetrieb an Schaltern oder ähnliche rechnergestützte Systeme im Bereich der Behörden. Sie gehören in der Mehrzahl in die kommerzielle, vorwiegend zeichenreihenorientierte Datenverarbeitung.

Systeme der genannten Art werden häufig "DB/DC-Systeme" (für: Datenbank-/Datenkommunikationssystem) genannt. Dabei bilden allerdings diese Systeme meist erst mit einem Allzweck-Betriebssystemkern zusammen eine Ablaufumgebung der Art, wie sie in dieser Norm beschrieben wird. Solche internen Gliederungsmerkmale der Ablaufumgebung realisierenden Programmsysteme sind für die Aussagen dieser Norm grundsätzlich unwesentlich.

Die in dieser Norm zu definierenden Funktionen eines DB/DC-Systems bilden die dem DC-Teil zuzurechnende Teilgruppe. Dieser Teil eines DB/DC-Systems wird in manchen Zusammenhängen auch als ein "TP-Monitor" (für: Teleprocessing-Monitor) oder als ein "Transaktionsmonitor" bezeichnet.

Da allgemein anerkannte und an allgemein zugänglicher Stelle präzise beschriebene Begriffsbildungen, die für die folgende Beschreibung benötigt werden, fehlen, ist ein einleitender, Begriffe klärender Teil unvermeidbar.

In den folgenden drei Abschnitten 0.2 bis 0.4 werden zunächst die Merkmale eines transaktionsorientierten Anwendungssystems allgemein beschrieben. Eine Einengung der Darstellung auf den KDCS-TAS-Kern erfolgt erst danach.

0.2 Externe Charakterisierung eines transaktionsorientierten Anwendungssystems

Ein transaktionsorientiertes Anwendungssystem, ist ein Teilhaberrechnungssystem*, d.h. ein rechnergestütztes Vielbenutzersystem.

Die physische Basis für die Schnittstelle* zwischen einem TAS und seiner Umwelt und damit vor allem zu seinen Benutzern ist ein Bestand an speziellen Eingabe- und Ausgabeeinheiten*. Dieser Bestand umfaßt vor allem Benutzerstationen* für die Eingabe und Ausgabe im Dialogbetrieb* (Dialogdatenstationen) sowie nur

0 Anwendungsbereich und Zweck

Vorbemerkung zur Schreibweise:
ein hochgestellter Stern unmittelbar hinter einem Wort (Beispiel: Schnittstelle*) verweist auf eine Definition in einer der DIN-Begriffsnormen DIN 44 300, DIN 66 200 Teil 1, DIN 44 302. Der Verweis wird nur beim ersten Auftreten des Wortes gegeben.

Liste von Abkürzungen (es werden hier nur solche Abkürzungen erklärt, die in der verbalen Beschreibung häufig verwendet werden):

AP	Anwendungsprogramm
DP	Dialogprogramm
DPL	Dialogprogrammlauf
DSB	Datenstationsspeicherbereich
FP	freilaufendes Programm
FPL	freilaufender Programmlauf
GSSB	Globaler Sekundärer Speicherbereich
KA	TAS-Kernaufbau
KB	Kommunikationsbereich
LSSB	Lokaler Sekundärer Speicherbereich
PL	Programmlauf
SPAB	Standard Primärer Arbeitsbereich
TAS	transaktionsorientiertes Anwendungssystem

0.1 Einleitung

Diese Norm definiert Wirkung und Syntax von Funktionen eines sehr speziellen Systemkerns, des sog. KDCS-TAS-Kerns, der eine Ablaufumgebung für die Programme eines transaktionsorientierten Anwendungssystems (Kurzform: TAS) realisiert. Die Benennung "KDCS-TAS-Kern" soll einen Bezug zu den kompatiblen Schnittstellen für Datenkommunikationssysteme (KDCS) herstellen. Wegen weiterer Details vgl. die historischen Anmerkungen in den Er-

- 5 -

für die Ausgabe geeignete Datenstationen (Ausgabedatenstationen). Die Merkmale und die Orte der beteiligten Rechensysteme* sind auf der hier einzuhaltenden Abstraktionsebene unerheblich.

An den Dialogdatenstationen arbeiten Benutzer, zeitlich unabhängig voneinander, vorzugsweise im strengen Wechselgespräch mit dem TAS. Ein Ereignispaar aus der Übergabe einer Nachricht* von einer Datenstation an das TAS und der folgenden Ausgabe der Antwortnachricht an derselben Dialogdatenstation zusammen mit dem vom TAS aufgrund der Eingabenachricht bis zur Erzeugung der Ausgabenachricht erbrachten Leistungen (z.B. Veränderungen am globalen Datenbestand, vgl. unten) heiße eine Interaktion. Die durchschnittliche Dauer einer Interaktion ist auf einige Sekunden anzusetzen, ihre maximale Dauer auf wenige Minuten.

Bei einer Interaktion übergibt der Benutzer in Form einer Eingabenachricht einen Auftrag an das TAS. Der Inhalt eines solchen Auftrags im Dialog besteht stets aus dem Aufruf einer oder mehrerer Funktionen aus einem mit dem TAS fest vorgegebenen Funktionsbestand (TAS-Funktionen). Die Benutzer können in keiner Weise selbst gestaltend auf den Funktionsbestand des TAS einwirken. Der Funktionsbestand umfaßt typischerweise einige zehn bis einige hundert Funktionen.

Durch die Funktionsausführungen wird ein für das TAS charakteristischer globaler Datenbestand bearbeitet, d.h. gelesen und bei einem Teil der Zugriffe auch geändert. Der globale Datenbestand ist in Kollektionen von Sätzen gegliedert, wie beispielsweise Dateien und Datenbanken. Die TAS-Funktionen verändern den globalen Datenbestand nicht auf der Ebene der Satzkollektionen. Nur einzelne Sätze und ggf. Beziehungen zwischen einzelnen Sätzen können erzeugt, geändert und gelöscht werden. Die Charakterisierung des Datenbestandes als 'global' weist dabei auf die Benutzung durch viele gleichartige und weitgehend gleichberechtigte Instanzen hin und sie drückt Beständigkeit aus auch dann, wenn sich das Kollektiv der zugreifenden Instanzen - insbesondere anzahlmäßig - in der Zeit verändert.

Anmerkung: Für den eingeführten Begriff des globalen Datenbestands ist es unerheblich, von welchen Teilen des TAS er realisiert und verwaltet wird, beispielsweise vom DB-Teil oder vom DC-Teil - falls solche Einteilungen erkennbar sind. Der globale Datenbestand kann daher auch sehr heterogen sein, so daß man eher von mehreren globalen Datenbe-

- 6 -

ständen in einem TAS sprechen würde (etwa eine Datenbank neben einem Dateiensystem oder auch nur einigen speziellen Dateien).

Um Strukturmerkmale des globalen Datenbestandes oder um den Funktionsbestand zu ändern, muß ein TAS angehalten werden, d.h. es muß außer Betrieb gehen. Solche Änderungen sind in fertig entwickelten Systemen nur in längeren Zeitabständen zu erwarten.

Das Arbeiten eines TAS-Benutzers ist noch in andere, tendenziell größere Einheiten gegliedert als die Interaktionen: eine oder mehrerer konsekutive Interaktionen bilden eine Transaktion (in TAS mit einem KDCS-TAS-Kern ist stets eine einzige Interaktion bereits eine Transaktion), mehrere konsekutive Transaktionen bilden einen Dialog. Zu einem Dialog bildet das TAS einen streng lokalen Teilzustand, d.h. ein auf diesen Dialog bezogenes "Gedächtnis". Solche lokalen Ausführungsumgebungen sind jeweils für alle anderen Benutzer unsichtbar und unzugänglich.

Die Wirkungen einer Transaktion lassen sich in folgende Kategorien einteilen:

- 1) Manipulationen am globalen Datenbestand.
- 2) Einfügen von Datenobjekten in die Ausgabeströme zu den Ausgabedatenstationen (freilaufende Ausgabebefehle; vgl. weiter unten in diesem Abschnitt). Diese Wirkungen gelten als geschehen, wenn die Datenobjekte in die Ausgabeströme unwiderruflich eingefügt worden sind, d.h. nicht erst, wenn deren Ausgabe erfolgt ist.
- 3) Erzeugen von freilaufenden Aufträgen (vgl. weiter unten in diesem Abschnitt), die nicht einfach Ausgabebefehle sind.
- 4) Veränderungen in der lokalen Ausführungsumgebung.
- 5) Erzeugen einer Nachricht (oder eines Teils dieser Nachricht) für die zugeordnete Dialogdatenstation.

1), 2), 3) und 5) mögen naheliegenderweise die globalen Wirkungen der Transaktion heißen. Dies sind gerade solche Wirkungen, die selbst oder deren Folgewirkungen etc. vom System nicht mehr nachträglich - etwa bei Auftreten eines Fehlers - sicher annullieren.

liert werden können. Dies gilt auch für 5), obwohl der Benutzer eine in den Dialog eingebundene Instanz ist. Es kann aber nicht verhindert werden, daß er seine Informationen aus dem Dialog kurzfristig in nicht annullierbarer Weise an Dritte weitergibt.

Die in den einzelnen Transaktionen durch Funktionsaufrufe und -ausführungen erzielten Wirkungen (man denke insbesondere, aber nicht nur an die Leszugriffe und an Veränderungen am globalen Datenbestand) sind bei zeitlicher Überlappung mehrerer Transaktionen so, als wären diese Transaktionen in irgendeiner Reihenfolge nacheinander ausgeführt worden. Diese Regel möge Serialisierung heißen.

Zwischen zwei konsekutiven Transaktionen eines Benutzers können sich im Prinzip (d.h. soweit dies zeitlich möglich ist) alle Veränderungen am globalen Datenbestand vollziehen, die von anderen Benutzern überhaupt bewirkt werden können. Anders ausgedrückt: zwischen zwei Transaktionen kann ein Benutzer arbeiten und anderer Benutzer in keiner Weise behindern oder verhindern; zwischen zwei Transaktionen muß sich ein Benutzer völlig vom globalen Datenbestand "lösen". Deswegen wird hier die Bezeichnung "Loslösungsregel" verwendet.

Anmerkung: Komplementär gilt: Umfang und Dauer der Arbeiten innerhalb einer betrachteten Transaktion stehen in engem Zusammenhang mit dem Ausmaß der (zeitlichen) Behinderungen, die für parallel stattfindende Transaktionen von ihr möglicherweise ausgehen.

Können in einer Transaktion aus irgendeinem Grunde (sei er vom Benutzer zu verantworten wie fehlerhafte Parameterwerte, fehlende Zugriffsrechte, oder vom TAS zu verantworten wie defekte Baueinheiten* oder andere Fehlfunktionen) nicht alle beabsichtigten globalen Wirkungen erzielt werden, so gehen überhaupt keine globalen Wirkungen von dieser Transaktion aus. Diese Festlegung möge die Alles-oder-Nichts-Regel heißen.

Anmerkung: Falls die vom Benutzer gewünschten globalen Wirkungen einer Transaktion in einem Dialog definitiv nicht erzielt werden können, so wird der Benutzer i.a. durch eine Ausgabenachricht darüber informiert. Diese Ausgabenachricht gilt natürlich nicht als eine der beabsichtigten globalen Wirkungen der Transaktion.

Ein TAS führt auch Aufträge aus, bezüglich welcher ein Benutzer während ihrer Ausführung keine Nachrichten mit dem TAS austauschen kann, denen also insbesondere keine Dialogdatenstation zugeordnet ist. Diese Aufträge heißen hier freilaufende Aufträge, ihre Ausführungen im TAS freilaufende Auftragsabwicklungen*. Die Bezeichnung "freilaufend" soll ausdrücken, daß diese Auftragsabwicklungen niemals in Wartezustände mit Bezug auf Aktionen von Benutzern oder auf Aktionen anderer Auftragsabwicklungen eintreten, d.h. sie warten höchstens beim Zugriff auf globale Daten wegen zufälliger Zugriffskonkurrenz oder auf die Zuteilung anderer Betriebsmittel wie Speicherbereiche, Datenstationen etc. Auch freilaufende Aufträge sind inhaltlich ausschließlich mit Hilfe des vorgegebenen Funktionsbestandes des TAS aufzubauen.

Für jede freilaufende Auftragsabwicklung wird eine lokale Ausführungsumgebung - ähnlich wie für Dialoge - eingerichtet und mitgeführt.

Eine wichtige Untergruppe der freilaufenden Aufträge sind die freilaufenden Ausgabeaufträge. Mit ihnen können geeignete Datenobjekte in die Ausgabeströme zu den genannten Ausgabedatenstationen eingefügt werden. Sie entstehen oft als Nebenwirkungen aus Dialogen heraus.

Freilaufende Aufträge können direkt von allen eingabefähigen Datenstationen aus erteilt werden, sie können aber auch als Teilergebnis einer Interaktion oder auch aus freilaufenden Auftragsabwicklungen heraus entstehen.

Die Ausführungsdauer von freilaufenden Aufträgen ist typischerweise wesentlich länger als die der Interaktionen.

Auch die freilaufenden Auftragsabwicklungen sind in eine Folge von Transaktionen gegliedert, d.h. sie sind in nahtlos aufeinanderfolgende Teilabwicklungen gegliedert, für die jeweils die für Transaktionen angegebenen Regeln gelten. Dabei können wenig umfangreiche freilaufende Auftragsabwicklungen durchaus nur aus einer einzigen Transaktion bestehen (in TAS mit einem KDCS-TAS-Kern besteht jeder freilaufende Auftragsabwicklung aus genau einer Transaktion; umfangreiche nicht-dialogische Arbeiten können dann als Ketten aus vielen freilaufenden Auftragsabwicklungen realisiert werden).

Ein TAS erfüllt sehr hohe Anforderungen an Stabilität und Korrektheit seines Funktionierens. Die Benutzer dürfen dafür in keiner Weise zur Hilfestellung herangezogen werden.

Ein TAS kennt Zugangsregulierungen, die beispielsweise festlegen, welche Benutzer mit welchen Funktionen welche Teile des globalen Datenbestandes manipulieren dürfen. Dies setzt Prozeduren voraus, gemäß denen sich ein Benutzer vor dem System identifizieren kann.

0.3 Aufgaben eines TAS-Kerns

In diesem Abschnitt soll eine weitergehende Gliederung eines TAS in Instanzen* und einiges Grundsätzliche zu ihrem Zusammenwirken ausgesagt werden. Die auf dem Markt angebotenen DB/DC-Systeme sind zum Teil so verschieden, daß die folgende Darstellung keinesfalls für alle diese Produkte Gültigkeit haben kann; sie orientiert sich schwerpunktmäßig an dem zu normenden KDCS-TAS-Kern.

Der Funktionsbestand eines TAS wird durch einen Bestand von ladbaren Programmen konkretisiert. Diese Programme heißen die Anwendungsprogramme (AP) des TAS. Eine TAS-Funktionsauswertung wird dabei nur in Ausnahmefällen durch Ausführung eines einzelnen AP erbracht. Die AP-Läufe werden typischerweise zu Ketten verbunden, d.h. es werden keine hierarchischen Ablaufstrukturen gebildet.

Anmerkung: Die Bezeichnung "Programm" bedeutet im folgenden ausschließlich Programmtext, Programmkörper, d.h. ein syntaktisches Gebilde. Die Funktionseinheit, die durch die Ausführung eines Programms in einer geeigneten Umgebung entsteht, wird immer als "Programmlauf" bezeichnet.

Die höchststufige Gliederung des gesamten TAS umfaßt als Instanzen* den TAS-Kern, die EA-Einheiten wie in Abschnitt 0.2 eingeführt (jeweils mit ihren Bedienern/Benutzern als eine Einheit aufgefaßt) sowie die jeweils gerade existierenden AP-Läufe. Der TAS-Kern kommuniziert mit den Instanzen der beiden anderen Arten im strengen Wechselgespräch. Eine Kommunikation zwischen AP-Läufen und EA-Einheiten ohne Zwischenschaltung des TAS-Kerns findet nicht statt. AP-Läufe werden unter gewissen Bedingungen vom TAS-Kern initiiert.

Ein AP-Lauf ruft den TAS-Kern durch parametrisierten Aufruf jeweils einer sog. TAS-Kernfunktion auf, wonach er in einen Wartezustand eintritt, bis der TAS-Kern antwortet. Der Aufruf einer TAS-Kernfunktion wird kurz TAS-Kernauf genannt. Die TAS-Kernfunktionen zusammen mit Festlegungen über die jeweils zulässigen Konstellationen von Parameterwerten repräsentieren das explizit aufrufbare Dienstangebot des TAS-Kerns für die AP.

Ein AP-Lauf ist in eine Ausführungsumgebung eingebettet, die entweder einem Dialog oder der Abwicklung eines freilaufenden Auftrags zugeordnet ist. In diesen Ausführungsumgebungen laufen i.a. nacheinander mehrere AP-Läufe ab. In den Phasen zwischen zwei aufeinanderfolgenden AP-Läufen innerhalb einer Ausführungsumgebung muß diese vollständig vom TAS-Kern erhalten werden. Die Gesamtheit aller systemseitigen Aktivitäten für einen Dialog bzw. für eine Abwicklung eines freilaufenden Auftrags, d.h. das Bilden und Erhalten einer Ausführungsumgebung, diverse AP-Läufe in dieser Ausführungsumgebung, der Vollzug der TAS-Kernaufträge aus diesen AP-Läufen durch den TAS-Kern, heißen ein Vorgang. Man unterscheidet Dialogvorgänge und freilaufende Vorgänge.

Der TAS-Kern umfaßt aus der Sicht der AP-Läufe sowohl den globalen Datenbestand als auch den Bestand an Anwendungsprogrammen; denn jeglicher Zugriff auf diese beiden Bestände (Zugriff auf Sätze; Start von Programmen) ist für die AP-Läufe ausschließlich in der Form von TAS-Kernaufrufen möglich.

Funktionen für Zugriffe zum globalen Datenbestand werden hier im Detail nicht behandelt, es sei denn, sie beziehen sich auf solche globalen Dateien, die als Sondereinrichtungen des DC-Teils aufgefaßt werden können.

Die auf AP gestützten Tätigkeiten innerhalb einer Interaktion bestehen immer aus einer Kette vollständiger AP-Läufe. Dasselbe gilt für die Ausführung eines freilaufenden Auftrags (in TAS mit einem KDCS-TAS-Kern: genau ein AP-Lauf). Eine Konsequenz dieser Festlegung ist, daß innerhalb eines AP-Laufs nicht auf eine Eingabenachricht von einer Dialogdatenstation gewartet werden kann. Ein solcher Wartezustand ist systemseitig nur zwischen zwei AP-Läufen möglich.

Anlässe für den TAS-Kern, einen AP-Lauf zu starten, sind folgende:

- Eintreffen einer Nachricht von einer Dialogdatenstation.
- Endemeldung eines AP-Laufs innerhalb einer Interaktion, ohne daß damit das Interaktionsende erreicht ist.
- Vorliegen eines ausführbaren freilaufenden Auftrags und Erfülltheit aller übrigen (z.B. betriebsmitteltechnischen) Bedingungen für den Start seines ersten AP-Laufs.

Die Kenntnis, welches AP jeweils zu starten ist, gewinnt der TAS-Kern zum Teil aus den Eingabenachrichten, zum Teil aus Angaben der vorher beendeten AP-Läufe, wobei in Grenzfällen der eine oder der andere Teil fehlen kann.

Der Nachrichtenaustausch zwischen den AP-Läufen und den EA-Einheiten wird - wie bereits gesagt - ausschließlich vom TAS-Kern vermittelt, wobei die AP von allen Details eines eventuell in das Gesamtsystem einbezogenen eigenständigen Datenübermittlungssystems völlig abgesichert werden. Die EA-Einheiten haben aus der Sicht der AP-Läufe TAS-spezifische Namen.

Der Nachrichtenaustausch wird immer durch Zwischenpufferung im TAS-Kern bewerkstelligt. Eine Eingabenachricht wird von einem AP-Lauf, eventuell in mehreren Schritten, beim TAS-Kern abgerufen. Entsprechendes gilt für Ausgabenachrichten und für freilaufende Ausgabeaufträge.

Eine wichtige Leistung des TAS-Kerns ist es, den Nachrichtenaustausch auf der Ebene angemessen abstrakter Gerätemodelle zu unterstützen. Dies geschieht für die Dialogdatenstationen u.a. durch die Hinterlegung von vordefinierten Bildschirmanteilen (Datenstationsformaten) und entsprechenden Nachrichtenformaten, die in einer Bibliothek beim TAS-Kern gehalten und von den AP-Läufen benannt und benutzt werden.

Der TAS-Kern etabliert und erhält die lokalen Ausführungsumgebungen für Dialogvorgänge und für freilaufende Vorgänge über deren gesamte Dauer hin. Die zu einer Ausführungsumgebung lokalen, also zu einem Vorgang gehörenden, aber einen AP-Lauf überlebenden Datenobjekte teilen sich in verschiedene Arten, z.B.:

- langlebige Datenobjekte außerhalb der AP-Läufe, z.B. Dateien.
- kurzlebige Datenobjekte außerhalb der AP-Läufe, die von einem AP-Lauf beim TAS-Kern hinterlassen und genau vom nächsten wieder abgerufen werden können ("Nachrichtenübergabe entlang der AP-Lauf-Kette").
- Vererbung von AP-Lauf-lokalen Datenobjekten ("Arbeitsspeicher-/Arbeitsspeicher-Transport"), d.h. ein AP-Lauf findet vorbereitete lokale Datenobjekte vor, deren Vorbereitungen aus korrespondierenden Datenobjekten des unmittelbar vorher beendeten AP-Laufs stammen. Diese Datenobjekte sind de facto langlebiger, obwohl sie als lokale Datenobjekte der AP-Läufe dargestellt werden.

Die in einem TAS zusammenarbeitenden Instanzen müssen sich über Beginn und Ende von Transaktionen verständigen. Dies geschieht im Dialogbetrieb aus Benutzersicht dann implizit, wenn es nur Transaktionen aus genau einer Interaktion gibt. Ein AP-Lauf in einer Dialogumgebung muß ggf. dem TAS-Kern explizit signalisieren, daß mit seinem Abschluß auch der Abschluß der laufenden Transaktion erreicht wird.

Der TAS-Kern sorgt für die Einhaltung der Serialisierungs- und der Lösungsregel. Dies wird hinsichtlich der Arbeiten am globalen Datenbestand üblicherweise durch globale Sperren (zugunsten einzelner Transaktionen) auf betroffenen Teilen des globalen Datenbestandes erreicht, die bei Transaktionsende auf jeden Fall alle aufgehoben werden.

Die anderen globalen Wirkungen einer Transaktion (Hervorbringen freilaufender Aufträge, Ausgeben einer bereits beim TAS-Kern vorliegenden Nachricht an die Datenstation) müssen auch wegen der Alles-oder-Nichts-Regel mindestens bis zum Ende der betreffenden Transaktion beim TAS-Kern gepuffert werden, d.h. mit ihrer Ausführung darf bis dahin nicht begonnen werden. Dies wird alles ohne Mitwirkung der AP gewährleistet.

Die Arbeitsweise des TAS-Kerns soll ferner die AP-Läufe von der Behandlung vom TAS-Kern zu verantwortender Fehler vollständig absichern, d.h. in den AP erscheinen keinerlei Anweisungen, die zu einer solchen Fehlerbehandlung irgendeinen Bezug haben. Für die Vorgehensweise des TAS-Kerns ist es naheliegend, die Transaktionen gleichzeitig zu Sicherungsabschnitten zu machen, d.h. am Beginn einer Transaktion Vorkehrungen zu treffen, daß

etwa ein betroffener Dialog auf den Zustand bei Beginn dieser Transaktion zurückgesetzt werden und wieder anlaufen kann.

Die diversen Vorkehrungen, die eine Transaktion zum Sicherheitsabschnitt machen, versetzen den TAS-Kern auch in die Lage, für vom TAS-Kern zu verantwortende und zu erkennende Fehler die Einhaltung der Alles-oder-Nichts-Regel zu garantieren. Man kann den AP eine spezielle TAS-Kernfunktion anbieten (Annullieren der bisher in einer Transaktion definierten und beim TAS-Kern hinterlegten globalen Wirkungen), die den AP ihren Teil an der Realisierung der Alles-oder-Nichts-Regel erleichtert (diese Funktion ist nicht im KDCS-TAS-Kern).

Anmerkung: Falls DB- und DC-Teil eines TAS-Kerns nur lose zusammenarbeitende Instanzen sind, so ist insbesondere das Bilden von Transaktionen ein wichtiger und möglicherweise komplizierter Teil ihres Zusammenwirkens. Dies gilt vor allem auch für die Eigenschaft der Transaktionen, gleichzeitig Sicherungsabschnitte zu sein.

Für diese Norm wird die volle Integration der beiden Funktionsgruppen des TAS-Kerns - der DB-orientierten und der DC-orientierten - als a priori gegeben vorausgesetzt, d.h. auf die erwähnten Abstimmungsprobleme wird im folgenden in keiner Weise mehr Bezug genommen.

Unter dem DB-Teil eines TAS-Kerns ist hier ggf. auch ein Dateiverwaltungssystem oder eine andere, Sätze speichernde Instanz (beispielsweise mit einer Schnittstelle nach DIN 66 263) zu verstehen, die nicht die Funktion eines Datenbanksystems im engeren Sinne erfüllen.

0.4 Der Begriff der TAS-Definition

Die Normung von Funktionen und anderen Leistungsmerkmalen eines TAS-Kerns hat als einen wichtigen Zweck, daß verschiedene Teile eines TAS von verschiedenen, weitgehend entkoppelt arbeitenden Institutionen beigebracht und zusammengefügt werden können. Eine naheliegende Arbeitsteilung ist folgende:

Eine Institution definiert die das TAS in seinem individuellen Inhalt bestimmenden Teile, nämlich

- den globalen Datenbestand (und eventuell eine Anfangsbelegung)

- die TAS-Funktionen (vgl. Abschnitt 0.2), repräsentiert durch einen Bestand an Anwendungsprogrammen
- Namen von Datenstationen mit Typangaben
- Datenstationsformate und Nachrichtenformate

Die andere Institution liefert die systemtechnisch-orientierten Anteile; sie

- hält alle übrigen Teile des TAS-Kerns vorgefertigt bereit. (Diese Teile sind von den erstgenannten individuellen Bestandteilen in dem Sinne unabhängig, daß sie für viele - inhaltlich verschiedene - TAS identisch sind)

Darüberhinaus ist festzulegen, in welcher Gestalt die Anwendungsprogramme vorzugeben sind, wie ihre im TAS-Betrieb zu benutzenden Namen angegeben werden, welche Datendefinitionssprache für den globalen Datenbestand benutzt werden kann, wie Datenstationsformate und Nachrichtenformate dargestellt, benannt und miteinander in Beziehung gesetzt werden, und manches mehr.

Das systematische Zusammentragen und Zusammenfügen der verschiedenen erwähnten textlichen Materialien, die schließlich einen startfertigen TAS-Text ergeben, heiße eine TAS-Definition.

In dieser Norm wird zwar über die syntaktische Ausgestaltung der erwähnten Definitionsbestandteile keine Festlegung getroffen. Es wird aber festzulegen sein, welche Merkmale (Parameter) eines betriebsbereiten TAS-Kerns in der TAS-Definition angegeben werden können und müssen. Aus diesem Grund sind noch einige Begriffe aus dem Umfeld der TAS-Definition einzuführen.

Die Definition eines TAS-Kerns bezieht sich auf ein bestimmtes Wirtssystem, das als gegeben vorauszusetzen ist. Dies ist üblicherweise ein Rechensystem* oder eine Rechenanlage*. Das Wirtssystem umfaßt insbesondere einen Bestand an EA-Einheiten, auf die man sich in der Definition des TAS beziehen kann. Ein Teil dieses Bestandes wird in der TAS-Definition zu EA-Einheiten des TAS erklärt, indem ihnen u.a. Namen zugeordnet werden, unter denen sie im TAS angesprochen werden können.

Das Ergebnis einer TAS-Definition ist (i.a. nach rechnergestützter Transformation der manuell vorbereiteten Texte) ein startfertiger TAS-Text, kurz: ein TAS-Text, der auf ein ganz bestimmtes konkretes Wirtssystem zugeschnitten ist.

Ein TAS-Text kann in einer Aktion (Systemstart) dergestalt mit dem Wirtssystem vereinigt werden, daß danach ein betriebsbereites TAS existiert, das die definierten Komponenten (EA-Einheiten, globaler Datenbestand, Bestand an Anwendungsprogrammen) enthält.

In dieser Norm wird die Bezeichnung TAS niemals für ein rein syntaktisches Gebilde (wie es ein TAS-Text ist) verwendet. Ein TAS ist hier stets eine agierende oder reagierende Funktionseinheit*, spezieller: eine Instanz.

Anmerkung: In einem Wirtssystem können mehrere TAS gleichzeitig existieren. In dieser Norm wird die Möglichkeit, daß solche koexistierenden TAS untereinander (innerhalb des Wirtssystems) Daten austauschen können, nicht in Betracht gezogen.

1 Allgemeines zur Beschreibung, Bezeichnung

1.1 Allgemeines zur Beschreibung

In Abschnitt 2 findet sich eine verbale Beschreibung, in Abschnitt 3 eine formale Beschreibung der Schnittstellen eines KDCS-TAS-Kerns. Die formale Beschreibung ist in einigen Aspekten der Schnittstellenfestlegungen weniger detailliert als die verbale Beschreibung oder die Inhalte der Tabellen (Abschnitt 4). Dies betrifft vor allem das Modell der Dialogdatenstation, aber auch die speicherungstechnische Darstellung der Datenobjekte, die zwischen dem KDCS-TAS-Kern und den Anwendungsprogrammläufen ausgetauscht werden. In jedem Fall stellt die jeweils weitergehende Detaillierung die Aussage der Norm dar.

1.2 Bezeichnung

Ein TAS-Kern, der Schnittstellen gemäß dieser Norm hat, wird wie folgt bezeichnet:

KDCS-TAS-Kern DIN 66 265-A

2 Verbale Beschreibung des KDCS-TAS-Kerns

INHALT des Abschnitts 2	Seite
2.1 Besonderheiten des KDCS-TAS-Kerns, Grundlegendes zu den Programmablauf-Schnittstellen	21
2.1.1 Besonderheiten des KDCS-TAS-Kerns, Terminologie ..	21
2.1.2 Grundlegende Details zur Parameterübergabe bei TAS-Kernauffufen	24
2.1.3 Kategorien von Statusanzeigen	29
2.1.4 Die Folge der TAS-Kernauffufe (INIT, PEND)	30
2.2 Die bei einer Systemdefinition einzubringenden Materialien und festzulegenden Parameter	31
2.3 Dialogvorgänge und die DPL-Schnittstelle	37
2.3.1 Anforderung der Eröffnung und Grobstruktur von Dialogvorgängen	37
2.3.1.1 Anforderung der Eröffnung eines Dialogvorgangs	37
2.3.1.2 Grobstruktur eines Dialogvorgangs (PEND)	40
2.3.2 Programmablauflokale Speicher	41
2.3.2.1 Der Arbeitsspeicher (INIT)	41
2.3.2.1.1 Grundsätzliches	41
2.3.2.1.2 Der Kommunikationsbereich (INIT)	42
2.3.2.1.3 Primäre Arbeitsbereiche (INIT, GTMM, FRMM) ..	44
2.3.2.1.4 Ergänzende Bemerkungen zum INIT-Aufruf (INIT)	46
2.3.2.2 Der Lokale Sekundäre Speicherbereich (LSSB) als programmablauf-lokaler Speicher	47
2.3.3 Systeminterne Datenübergabe von Programmablauf zu Programmablauf	47
2.3.3.1 Vorwangsinterne Datenübergabe	47
2.3.3.1.1 Datenübergabe im Kommunikationsbereich	47
2.3.3.1.2 Nachrichtentübergabe (MPUT, MGET)	47
2.3.3.1.3 Der Lokale Sekundäre Speicherbereich (LSSB) eines Dialogvorgangs (SPUT, SGET, SREL)	50
2.3.3.2 Vorgangsgrenzen überschreitende Datenübergabe /Eingangsdaten für freilaufende Vorgänge (FPUT, FGET)	52
2.3.4 Globale Speicher (SPUT, SGET, SREL, PTDA, GTDA) ..	54
2.3.4.1 Grundsätzliches zur Funktionsweise der globalen Speicher	54
2.3.4.2 Der Globale Sekundäre Speicherbereich (SPUT, SGET, SREL)	55
2.3.4.3 Der Datenstationsspeicherbereich des Dialogvorgangs (PTDA, GTDA)	56
2.3.5 Nachrichtenaustausch mit Datenstationen / Systemprotokoll	57
2.3.5.1 Datenstationstypen und Aufrufarten (MGET, MPUT, FPUT)	57
2.3.5.2 Nachrichtenaustausch mit der Dialogdatenstation des Vorgangs (MGET, MPUT, PEND-RE, PEND-FI) ...	59
2.3.5.2.1 Austausch formatierter Nachrichten	59
a) Einleitung	59
b) Die Detailstruktur der Formate und ihre Beziehungen in einer Formatgruppe	60

	Seite		Seite
c) Dynamische Abwicklung von Ausgabe und Eingabe	65	2.4.2 Programmlauflokale Speicher	89
d) Darstellung formatierter Nachrichten im Arbeitsspeicher	69	2.4.2.1 Der Arbeitsspeicher (INIT)	89
e) Mehrteilige Nachrichten	70	2.4.2.2 Der Lokale Sekundäre Speicherbereich (SPUT, SGET, SREL)	89
f) Details zu den zugehörigen TAS-Kern-Aufrufen	71	2.4.3 Systeminterne Datenübergabe von Programmlauf zu Programmlauf (FPUT, FGET)	89
2.3.5.2.2 Formatfreie Aufrufe und Aufrufe der FORMAT-Routine (MGET, MPUT)	73	2.4.4 Globale Speicher (SPUT, SGET, SREL, PTDA, GTDA) .	90
2.3.5.3 Freilaufende Ausgaben an Datenstationen (FPUT, PEND-RE, PEND-FI)	75	2.4.5 Nachrichtenaustausch mit Datenstationen / Systemprotokoll	90
2.3.5.4 Sonderfunktionen der Eingabe (MGET)	77	2.4.5.1 Eingabenachrichten (FGET)	90
2.3.5.5 Das Anwendungssystem-Protokoll (LPUT)	78	2.4.5.2 Ausgabenachrichten (FPUT)	90
2.3.6 Der Dialogschritt als Transaktion und Sicherungsabschnitt (PEND-RE, PEND-FI, PEND-ER) .	79	2.4.5.3 Das Anwendungssystem-Protokoll (LPUT)	91
2.3.6.1 Die Konsequenzen der Alles-oder-Nichts-Regel für die Arbeitsweise des TAS-Kerns	79	2.4.6 Der freilaufende Vorgang als Transaktion und Sicherungsabschnitt	91
2.3.6.2 Der Dialogschritt als Sicherungsabschnitt	84	2.4.7 Fehlerfreie Beendigung eines freilaufenden Vorgangs	92
2.3.6.3 Integriertes Datenbanksystem	85		
2.3.7 Fehlerfreie Beendigung eines Dialogvorgangs	85		
2.4 Freilaufende Vorgänge und die FPL-Schnittstelle ...	86		
2.4.1 Anforderung der Eröffnung und Grobstruktur eines freilaufenden Vorgangs	86		
2.4.1.1 Anforderung und Eröffnung eines freilaufenden Vorgangs (FGET, FPUT)	86		
2.4.1.2 Grobstruktur eines freilaufenden Vorgangs (PEND-FI)	88		

2.1 Besonderheiten des KDCS-TAS-Kerns, Grundlegendes zu den Programmlauf-Schnittstellen

2.1.1 Besonderheiten des KDCS-TAS-Kerns, Terminologie

Der KDCS-TAS-Kern ist ein TAS-Kern im Sinne von Abschnitt 0.3. Die Aussagen des Abschnitts 0.3 sind wesentliche Voraussetzung für die folgenden Darlegungen. Die wichtigsten Besonderheiten des KDCS-TAS-Kerns sind:

- in Dialogen bildet jede Interaktion eine Transaktion.
- ein (auf AP-Läufe gestützter) freilaufender Vorgang besteht aus genau einem Programmablauf eines AP. Er bildet genau eine Transaktion.
- die Menge der AP ist in zwei disjunkte Teilmengen gegliedert: die Dialogprogramme (DP) und die freilaufenden Programme (FP). Näheres zum Begriff des AP findet sich u.a. in Abschnitt 2.2.
- die KDCS-TAS-Kern/Programmlauf-Schnittstellen für die beiden Arten von Programmen sind nicht identisch. Andererseits gilt für die möglichen Inhalte von TAS-Kernauffrufen (Kurzform: KA), daß die Mehrzahl der TAS-Kernfunktionen an beiden Schnittstellen zur Verfügung steht.
- der KDCS-TAS-Kern enthält keine Datenbankfunktionen, bzw. diese sind nicht Bestandteil der hier im Detail festzulegenden Schnittstellen (vgl. Anmerkung am Ende von Abschnitt 0.3).

Der KDCS-TAS-Kern

- kennt in den Arbeitsspeichern der Programmläufe einige besondere Datenobjekte (Kommunikationsbereich, primäre Arbeitsbereiche).
- betreibt innerhalb von Dialogvorgängen die Weitergabe programmlokaler Datenobjekte von Arbeitsspeicher zu Arbeitsspeicher in Gestalt des Kommunikationsbereichs (vgl. insb. Abschnitte 2.3.2 und 2.3.3).

- unterhält vorgangslokale Satzspeicher mit Zugriff nur durch KA (vgl. Abschnitt 2.3.3/LSSB).
- unterhält vorgangsübergreifende Satzspeicher, einen systemglobalen (vgl. Abschnitt 2.3.4.2/GSSB) und einen pro eingeführten Datenstationsrufenamen (vgl. Abschnitte 2.2 und 2.3.4.3/DSB).
- setzt zur Parameterübergabe die Darstellung von Bytevektoren im Arbeitsspeicher der Programmläufe voraus (vgl. Abschnitt 2.1.2).

Da Verwechslungen nicht möglich sind, wird im folgenden ein Anwendungsprogramm einfach ein Programm genannt; entsprechend heißt es statt AP-Lauf nur P-Lauf bzw. PL. Weitere Abkürzungen sind DP/DPL für Dialogprogramm/Dialogprogrammablauf, FP/FPL für freilaufendes Programm/freilaufender Programmablauf.

Statt "KDCS-TAS-Kern" heißt es weiterhin nur "TAS-Kern".

Der TAS-Kern bildet nach außen zwei wesentlich verschiedene Arten von Schnittstellen, die Schnittstelle zu den Programmläufen und die Schnittstelle zu den (Benutzern an) Datenstationen. Der ganz überwiegende Teil der Norm befaßt sich mit der Schnittstelle zu den Programmläufen. Diese ist unterteilt in die DPL-Schnittstelle (vgl. Abschnitt 2.3) und die FPL-Schnittstelle (vgl. Abschnitt 2.4). Ist dieser Unterschied im jeweiligen Kontext nicht wesentlich, so wird nur von der PL-Schnittstelle gesprochen.

Im Abschnitt 2.3 resp. Abschnitt 2.4 wird die DPL-Schnittstelle resp. die FPL-Schnittstelle vollständig beschrieben, d.h. insbesondere, daß nicht als zulässig charakterisierte TAS-Kernaufforderungen unzulässig sind und zu Fehlermeldungen oder zum Abbruch des PL führen.

Dynamische Zuordnung der Datenstationen

An der PL-Schnittstelle werden zur Identifikation von Datenstationen Datenstationsrufenamen verwendet, die bei der Systemdefinition (vgl. Abschnitt 2.2) eingeführt werden. Die vom TAS-Kern jeweils zu realisierende Zuordnung von Datenstationsrufenamen zu

Datenstationen des Wirtssystems kann sich in der Laufzeit des TAS ändern. Zeitweilig können sogar eingeführte Datenstationsrufnamen ohne zugeordnete Datenstationen vorhanden sein.

Wird von einem PL beispielsweise zum Zeitpunkt t_1 eine frei laufende Ausgabenachricht N unter dem Datenstationsrufnamen drn an den TAS-Kern übergeben, so sind u.a. folgende Abläufe denkbar:

- bei t_1 ist drn die Datenstation dst zugeordnet, und zu einem Zeitpunkt $t_2 > t_1$ wird N an dst ausgegeben.
- bei t_1 ist drn keine Datenstation zugeordnet. N bleibt beim TAS-Kern liegen, bis drn eine Datenstation dst zugeordnet wird. Danach wird N an dst ausgegeben.
- bei t_1 ist drn die Datenstation dst₁ zugeordnet. Bevor N an dst₁ ausgegeben wird, wechselt die Zuordnung zur Datenstation dst₂. N wird nach diesem Wechsel an dst₂ ausgegeben.

Die Details der dynamischen Datenstationszuordnung zu den eingeführten Datenstationsrufnamen sind nicht Gegenstand der Norm.

Einem Datenstationsrufnamen wird bei seiner Einführung während der Systemdefinition eindeutig ein Typ ("formatfähig", etc.) und eine Betriebsart ('ein- und ausgabefähig', 'nur ausgabefähig') zugeordnet. Es wird im folgenden angenommen, daß diese Merkmale (Typ und Betriebsart) wenigstens temporär auch den zugeordneten Datenstationen zukommen. Eine Datenstation ist beispielsweise Formatdatenstation in der Zeit, in der sie über einen Datenstationsrufnamen mit dem Typ 'formatfähig' betrieben wird. Man wird eine Datenstation auch generell Formatdatenstation (in Bezug auf ein TAS) nennen, wenn für sie gelegentlich ein Betrieb als Formatdatenstation in Betracht kommt. Datenstationen sind keine Ausgabedatenstationen (in Bezug auf ein TAS), wenn sie bei seinem Betrieb ausschließlich über Datenstationsrufnamen mit der Betriebsart 'nur ausgabefähig' erreichbar sind.

Bei den Zuordnungen der Datenstationsrufnamen zu Datenstationen muß der TAS-Kern gewisse Einschränkungen beachten. Eine Datenstation, die gerätetechnisch überhaupt keine Eingaben zuläßt, kann nicht einem Datenstationsrufnamen mit der Betriebsart 'ein- und ausgabefähig' zugeordnet werden. Andererseits kann ein Bildschirmarbeitsplatz durchaus unter einem Rufnamen mit

der Betriebsartangabe 'nur ausgabefähig' erreicht werden; in diesem Betriebsfall ist er dann 'nur ausgabefähig'.

Als weiteres Beispiel der dynamischen Gerätezuordnung diene das Datenstationsbündel: mehrere reine Ausgabedatenstationen dst₁, dst₂, ..., dst_n werden zusammengefaßt. Diesem Bündel steht im TAS nur ein einziger eingeführter Datenstationsrufname drn mit der Betriebsart 'nur ausgabefähig' gegenüber. Zu jeder Zeit existiert genau eine Zuordnung von drn zu einer der Datenstationen des Bündels. So wird erreicht, daß der Begriff des Datenstationsbündels an der PL-Schnittstelle nicht in Erscheinung tritt.

Auch die dauerhafte Vor-Zuordnung einer Gruppe von eingeführten Datenstationsrufnamen zu genau einer Datenstation kann zweckmäßig sein. Je nachdem unter welchem Rufnamen die Datenstation angesprochen wird, können besondere Betriebsumstände vorliegen (Typ der Datenstation, Benutzerberechtigungen, voreingestellte Datenstationsformate etc.)

2.1.2 Grundlegende Details zur Parameterübergabe bei TAS-Kernaufrufen

Wie im Prinzip ein TAS-Kern mit den PL zusammenwirkt, wie beispielsweise die Übergabe eines Prozessors* in geordneter Weise erfolgt oder wie der TAS-Kern auf den Arbeitsspeicher eines PL zugreifen kann, wird als bekannt vorausgesetzt.

PL sind - auf der Detaillierungsebene üblicher Programmiersprachen betrachtet - sequentielle Prozesse. An der Schnittstelle zwischen einem PL und dem TAS-Kern wird ein TAS-Kernauftrag (KA) durch Übergabe des Prozessors von dem PL an den TAS-Kern eingeleitet. Der (ein) Prozessor wird frühestens vom TAS-Kern an den PL zurückgegeben, wenn die im KA angeforderte Funktion erbracht ist bzw. wenn feststeht, daß sie nicht erbracht werden kann, der PL aber trotzdem fortgesetzt werden kann.

Zu den Programmiersprachen, die zur Programmierung von Anwendungsprogrammen verwendet werden können, gibt es spezielle Anweisungen, deren Ausführung einen KA bewirkt. Sofern bei der programmiersprachlichen Gestalt eines solchen Aufrufs ein frei wählbarer Name zur Kennzeichnung des aufgerufenen Programms (hier: der TAS-Kern) auftritt, so ist dafür die Zeichenreihe KDCS zu verwenden.

- 25 -

Zu einem PL gehört ein Arbeitsspeicher, das ist ein Speicher zur Aufnahme von Datenobjekten, die durch Ausführung von Anweisungen in den PL direkt, d.h. ohne Mitwirkung des TAS-Kerns in Gestalt von KA, bearbeitet werden können. Der Zugriff auf den Arbeitsspeicher kann mittels Adressen oder mittels Namen (von aktuell existierenden Variablen) oder anders erfolgen.

Anmerkung: Der Arbeitsspeicher ist nicht mit dem Hauptspeicher* zu verwechseln. Ein Zusammenhang besteht nur insofern, als die Arbeitsspeicher der PL meist im Hauptspeicher der Rechenanlage realisiert werden.

Ein PL kann nicht auf die Arbeitsspeicher anderer PL zugreifen.

Wesentliche Voraussetzung für die festzulegenden Beziehungen zwischen TAS-Kern und PL ist, daß die Arbeitsspeicher der PL Vektoren mit je einem 8-stelligen Binärwort als Wert einer Komponente (Bytevektoren) aufnehmen können.

Die zur Realisierung der Bytevektoren verwendete Speicherabbildung kann von der Programmiersprache abhängen, in der das Quellprogramm des PL programmiert ist. Dieses Merkmal des PL ist dem TAS-Kern bekannt (vgl. Abschnitt 2.2 Systemdefinition. Stichwort: Programmattribute), so daß er die jeweilige Speicherabbildung berücksichtigen kann. Für jede Programmiersprache, in der AP programmiert werden können, ist daher - unabhängig von den Wirtssystemen - festgelegt, wie diese Bytevektoren mit den Mitteln der Programmiersprache dargestellt werden.

Anmerkung: Bytevektoren im Arbeitsspeicher implizieren nicht, daß der Hauptspeicher der Rechenanlage Speicherzellen* zur Aufnahme 8-stelliger Binärwörter hat.

Bei den besagten Bytevektoren, die zum Datenaustausch zwischen TAS-Kern und PL dienen, handelt es sich um einen Datentyp speziell für die PL-Schnittstelle; als solche müßten sie etwa K-Bytevektoren heißen. Darauf wird verzichtet, weil auf einen eventuell in den verwendeten Programmiersprachen (andere) definierten Typ Bytevektor nicht Bezug genommen wird.

Es wird die Schreibweise $A[1]$, $A[2]$, ... für die Komponenten bzw. Komponentenwerte eines Bytevektors A verwendet. Das Indexintervall beginnt stets mit Abschnitt 1. Für den Teilvektor $A[i]$, $A[i+1]$, ... $A[m]$, $1 \leq i \leq m$, wird auch $A[i..m]$ geschrieben.

- 26 -

Ein KA enthält ein oder zwei sog. unmittelbare Parameter: nämlich stets den PARAM1, eventuell zusätzlich den PARAM2. Der Wert eines unmittelbaren Parameters übermittelt dem TAS-Kern die Lage eines Bytevektors im Arbeitsspeicher des aufrufenden PL. Die Werte der unmittelbaren Parameter sind also Verweise*.

Im Falle des PARAM1 enthält der bezeichnete Bytevektor der Länge 40, der sog. Parameterbereich, in bestimmten Teilvektoren weitere Parameter des Aufrufs, sog. mittelbare Parameter (zur Lage, zur Länge, zu Benennungen und zur Art der Werte, vgl. Tabelle 2). Die Qualifikation "unmittelbar" oder "mittelbar" beim Begriff Parameter wird, wenn keine Konfusionsgefahr besteht, weggelassen. Die im Parameterbereich enthaltenen benannten Parameter dienen der Angabe

- der aufgerufenen TAS-Kernfunktion (Parameter sind das Funktionsmerkmal (kcfm) und die Funktionsausprägung (kcfa). Da der kcfm(kcfa)-Wert aus einer Folge von genau vier (zwei) Großbuchstaben besteht, ist die Schreibweise PEND-RE für eine TAS-Kernfunktion unmittelbar verständlich. In manchen Fällen gibt es keine Funktionsausprägung.)
- der Längen angeforderter Speicherbereiche oder der Länge des über PARAM2 erreichbaren Bytevektors (Parameter: kcdl, kcbl, kcnl, kcpl)
- von Programmrufrufen (Parameter: korn)
- von Datenstationsrufen bzw. Kennzeichen von Nachrichtenformaten (Parameter: kcdn, konk)
- von Ausgabefunktionen an Datenstationen (Parameter: kcdf)

Der PARAM2 übermittelt dem TAS-Kern die Lage des sog. Datenbereichs, eines Bytevektors, dessen Länge von KA zu KA wechseln kann. Diese Länge wird durch einen Parameter im Parameterbereich übermittelt. Ein Datenbereich wird benötigt, wenn vom PL zum TAS-Kern oder umgekehrt Daten übergeben werden müssen, deren Umfang nicht bereits durch die TAS-Kernfunktion des KA allein fixiert ist, beispielsweise Nachrichtenteile, Sätze globaler Speicher etc..

gegeben. Es wird angenommen, daß in jedem konkreten TAS mit einem TAS-Kern eine einzige solche Codierung verwendet wird. Die Portabilität der Anwendungsprogramme (d.h. ggf. ihrer Quellprogramme) wird dadurch erreicht, daß

- bei der Abfassung der Programme der genannte Zeichenvorrat an der PL-Schnittstelle benutzt wird - ohne Bezugnahme auf irgendeine (Binär-)Codierung.
- bei Ablauf der Programme für alle Übergaben von Zeichenreihen über die PL-Schnittstelle die systemspezifische Codierung durch 8-stellige Binärwörter benutzt wird. Dies wird wiederum durch eine für das konkrete TAS spezifische Anpassung der Programme ermöglicht, die die Bedeutung der Programme für den TAS-Zweck nicht verändert.

Hinweis: Werden in einem AP Zeichen oder Zeichenreihen miteinander bzgl. der Relationen "größer" oder "kleiner" verglichen, so können diese Relationen durch Rückgriff auf die in der verwendeten Codierung den Zeichen zugeordneten Zahlwerte (genauer: Zahlwerte der zugeordneten Binärwörter) definiert sein. Dann werden sich i.a. bei unterschiedlichen Codierungen unterschiedliche Werte für dieselben Vergleichsausdrücke ergeben, was die Portabilität der Programme in Frage stellt.

Eine herausgehobene Bedeutung als Vorgesetzterwert oder zur neutralen Verlängerung von Zeichenreihen hat das Zeichen "Zwischenraum", auch Leerzeichen genannt. Es wird angenommen, daß jedes verwendete Alphabet genau ein solches Zeichen hat, das bei der Ausgabe der Vorwärtsbewegung der Schreibereinheit um einen Schreibschritt dient.

Im Kommunikationsbereich und in den Parameterbereichen gibt es Längenangaben (in Byte), die ihrerseits in zwei aufeinanderfolgenden Bytepositionen, d.h. in 16 Binärstellen darzustellen sind. Diese Längenangaben müssen im sogenannten

- zulässigen Zahlbereich für Längenangaben (das ist das Intervall der ganzen Zahlen von 0 bis 32767) liegen.

Die Art der Zahldarstellung in den verfügbaren 16 Binärstellen kann abhängig von der Quellsprache Q sein, in der ein AP geschrieben ist. Die Quellsprache des AP ist dem TAS-Kern bekannt (vgl. Abschnitt 2.2, Unterabschnitt Anwendungsprogramme).

Beim Datenaustausch zwischen TAS-Kern und einem PL anläßlich eines KA wird ferner der sog. Kommunikationsbereich (KB) des PL benutzt. Der KB ist ein Bytevektor (Länge > 116), der stets im Arbeitsspeicher eines PL existiert (weitere Einzelheiten zu seinem Aufbau und seiner Verwendung in den Abschnitten 2.3.1.1 und 2.3.2.1.2) und dessen Lage dem TAS-Kern a priori und für die Dauer des PL bekannt ist, d.h. insbesondere, daß unmittelbare Parameter bezüglich der dauerhaft fixierten KB-Teile entfallen. Der KB ist vollständig in drei disjunkte Bytevektoren zerlegt: den KB-Kopfbereich (Länge 84, Kurzbezeichnung: KB-Kopf), den KB-Rückgabebereich (Länge 32) und den KB-Programm-bereich (Länge > 0) in dieser Reihenfolge.

Im KB-Kopf darf durch Anweisungen des PL nur gelesen werden. Er enthält Grunddaten über den Vorgang und den PL, beispielsweise die Startzeit des Vorgangs und die des aktuellen PL (in einem Dialogvorgang), den Datenstationsrufenamen, den Programmrufoamen und andere mehr (für eine vollständige Übersicht vgl. Tabelle 4). Der TAS-Kern kann bei einem KA Parameterwerte für die Ausführung der aufgerufenen Funktion aus dem KB-Kopf entnehmen.

Im KB-Rückgabebereich liefert der TAS-Kern bei Ausführung eines KA einige Ergebnisparameter; darunter ist stets die Statusanzeige (kcsa) und ggf. eine Längenangabe (kcla), nämlich falls eine Bytefolge als Teilergebnis im Datenbereich geliefert wurde. Für eine vollständige Übersicht der Untergliederung des KB-Rückgabebereichs vgl. Tabelle 5 in Abschnitt 4.

Regel: (KB-Rückgabebereich nach KA)

Bei Rückkehr in einen PL nach einem KA haben alle Bytepositionen des Rückgabebereichs, die nicht der Darstellung von Ergebnissen dieses KA dienen, undefinierte Werte.

Regel: (niemals Änderungen im Parameterbereich)

Bei Rückkehr in einen PL nach einem KA sind niemals irgendwelche Datenwerte im Parameterbereich verändert.

An der PL-Schnittstelle und auch an der Schnittstelle zu den Datenstationen treten häufig Parameterwerte auf, die Zeichenreihen sind. Diese Zeichenreihen sind vorzugsweise über dem Zeichenvorrat (Alphabet) des 7-Bit-Codes nach DIN 66 003, Ausgabe Juni 1974, CODE-Tabelle 2, zu bilden, und zwar über dem Ausschnitt aus den Spalten 2 bis 7. Eine Codierung der Zeichen als 8-stellige Binärwörter ist zwar beim Betrieb an der PL-Schnittstelle unabdingbar, wird jedoch in dieser Norm nicht an-

Außerdem kann diese Zahldarstellung von dem konkreten TAS-Kern und dem Wirtssystem – beispielsweise von dem eingesetzten Compiler – abhängen. Die Fehlermeldung "Längenangabe ungültig" bedeutet dann, daß im Sinne der jeweils gültigen Darstellungskonventionen das angebotene 16-stellige Binärwort keine Zahl aus dem zulässigen Zahlbereich darstellt.

Wie beim Alphabet dürfen sich die oben genannten Abhängigkeiten aber nicht auf die Programme und die Programmierung in Q auswirken. Eine Norm über die Programmierung von AP für TAS in der Programmiersprache Q soll u.a. Festlegungen über die Gestalt von TAS-Kernauffufen in Q sowie der programmseitigen Zugriffe auf KB-Inhalte enthalten, und diese Festlegungen dürfen ihrerseits nicht von dem konkreten TAS-Kern und auch nicht von dem Wirtssystem sondern ausschließlich von der Programmiersprache Q abhängen. Für die Längenangaben bedeutet dies, daß entweder ein Datentyp von Q dafür verwendet werden kann oder daß es im Rahmen der fraglichen Norm festgelegte Transferprozeduren gibt.

2.1.3 Kategorien von Statusanzeigen

Die Werte, die nach einem KA als Statusanzeige kcsa auftreten können, sind in sechs Kategorien eingeteilt. Eine von diesen zeigt bei Eingaben von Datenstationen an, daß eine Funktionstaste betätigt wurde (vgl. Abschnitt 2.3.5.4). Sie gehört nicht in den Zusammenhang der Anzeigen von Irregularitäten. Die verbleibenden Kategorien sind:

- Anzeige ordnungsgemäßer Ausführung ("fehlerfrei").
- Bemerkungen: die geforderte Funktion wurde in eingeschränkter Weise ausgeführt.
- Warnungen: die geforderte Funktion wurde nicht durchgeführt. Nach Durchführung geeigneter Maßnahmen kann der PL sinnvoll fortgesetzt werden.
- Fehler: die geforderte Funktion wurde nicht durchgeführt. Vermutlich ist der PL nicht sinnvoll fortsetzbar.
- schwerwiegende Fehler: eine Fortsetzung des PL ist definitiv nicht möglich. Er wird durch den TAS-Kern beendet. Verschiedene Statusanzeigewerte sind trotzdem definiert, weil sie in den Programmen, die nach der Fehlererkennung bei der Ausführung

zung der Vorgangsbeendigung zum Ablauf kommen, benutzt werden. Zu diesen Programmen gehört insbesondere die sog. EXIT-Routine, ein systemspezifisches, nicht zu den genormten Leistungen des TAS-Kerns gehörendes Programm. Die Beendigung von Vorgängen im Fehlerfall ist im Detail in den Abschnitten 2.3.6.1 und 2.4.6 beschrieben.

Regel: (Auswahl unter mehreren möglichen Statusanzeigewerten) Sind mehrere Werte der Statusanzeige möglich, so wird unter diesen möglichen Werten diejenige aus der höchsten Kategorie (im Sinne der obigen Auflistungsreihenfolge) gewählt. Innerhalb dieser Kategorie wird der niedrigste mögliche Wert ausgewählt.

Für eine vollständige Übersicht über die möglichen Werte des Parameters kcsa, ihre Hauptbedeutungen und ihre Zuordnung zu den Kategorien vgl. Tabelle 7 in Abschnitt 4.

Weitere Angaben zu Fehlern finden sich bei den Detailerörterungen der einzelnen TAS-Kernfunktionen. Fehler, die bei vielen verschiedenen Arten von TAS-Kernauffufen gleichermaßen vorkommen können – wie etwa: unzulässiges Funktionsmerkmal oder ungültige Längenangabe oder fehlender bzw. ungültiger Verweis auf den Datenbereich – werden hingegen nicht in den Detailerörterungen der einzelnen TAS-Kernfunktionen besprochen. Angaben zu ihnen finden sich in Tabelle 7 in Abschnitt 4.

2.1.4 Die Folge der TAS-Kernauffufe (INIT, PEND)

Regel: (Erster und letzter TAS-Kernauffuf) Bezeichnet man mit ka_1 , ka_2 , ..., ka_n die nach der Zeit geordnete Folge aller KA eines PL, so muß ka_1 als einziger der ka_i das Funktionsmerkmal INIT (aus engl. initialization), ka_n als einziger der ka_i das Funktionsmerkmal PEND (aus engl. program end) haben.

Abweichungen von dieser Regel sind schwerwiegende Fehler (vgl. Abschnitt 2.1.3).

Über die Reihenfolge der ka_2 , ..., ka_{n-1} lassen sich allein auf Grund der aufgerufenen Arten von TAS-Kernfunktionen, also auf Grund von $kcfm$ - und $kfca$ -Wert, keine weiteren Regeln aufstellen.

Nach einem PEND-Aufruf (Programmlaufende) wird der PL niemals fortgesetzt. Ein PL kann natürlich auf Grund schwerwiegender Fehler auch ohne Auftreten eines PEND-Aufrufs beendet werden.

Die sonstigen Wirkungen eines INIT-Aufrufs (Programmlaufinitialisierung) und der diversen PEND-Aufrufe (es gibt mehrere Funktionensausprägungen) werden an anderen Stellen beschrieben.

2.2 Die bei einer Systemdefinition einzubringenden Materialien und festzulegenden Parameter

Mit Systemdefinition wird hier für den TAS-Kern der Begriff bezeichnet, der in Abschnitt 1.4 für ein TAS allgemein als 'TAS-Definition' eingeführt wurde.

Im folgenden werden die Arten von Materialien und die Parameter beschrieben, die bei einer Systemdefinition vor- bzw. festgelegt werden müssen. Unter dem Begriff 'Parameter' werden dabei auch verschiedene Zuordnungen (Funktionen) subsumiert.

Eine mehrfach benötigte Art von Werten ist die als Rufnamen bezeichnete Menge von Zeichenreihen der Länge 8, mit folgenden Eigenschaften:

- das erste Zeichen ist ein Buchstabe
- die Zeichenreihe hat ein Anfangsstück der Länge h , $1 \leq h \leq 8$, das nur aus Buchstaben oder Ziffernzeichen besteht.
- ist die Länge h des genannten Anfangsstücks kleiner als 8, so sind die Zeichen der Positionen $h + 1, \dots, 8$ Leerzeichen.

Anwendungsprogramme

Die im Hauptspeicher des Wirtssystems ablaufähigen Programmkörper aller Anwendungsprogramme sind beizubringen. Dies geschieht dadurch, daß zusammengesetzte Datenobjekte vom Typ TAS-Anwendungsprogramm gebildet werden, die aus einem Rufnamen, einem ablaufähigen Programmkörper, einer Typangabe (DP oder FP), Längenangaben und andere Angaben zum Kommunikationsbereich und zum Standard Primären Arbeitsbereich, sowie einer Angabe zur Quellsprache bestehen (detailliertere Auflistung weiter

unten). Ein TAS-Anwendungsprogramm wird auch einfach ein Anwendungsprogramm oder ein Programm genannt. Dieser Programmbegriff ist etwas spezieller als der in DIN 44 300, da er Angaben für die Betriebsumgebung umfaßt.

Der Rufname p kennzeichnet ein Programm eindeutig in einem TAS. Die in Programmobjekten verwendeten Rufnamen heißen daher die eingeführten Programmnamen des TAS. Man kann nunmehr von dem Programmkörper von p , dem Typ von p sprechen und die Funktionsschreibweise $\text{Programmkörper}(p)$, $\text{Typ}(p)$ usw. verwenden. Es werden verkürzte Sprech- und Schreibweisen wie "Start des Programms p " benutzt, wo es genauer "Start des Programmkörpers (p)" heißen müßte. Die Wendung "Programmlauf über p " ist so zu verstehen: Programmlauf über dem Programmkörper(p), wobei der Aufruf mittels des Programmnamens p erfolgte. Der beim Aufruf verwendete Rufname ist wichtig, da $\text{Programmkörper}(p_1) = \text{Programmkörper}(p_2)$ und doch $p_1 \neq p_2$ sein kann. Dabei kann sogar noch $\text{Typ}(p_1) \neq \text{Typ}(p_2)$ sein. Auch andere Merkmale der Programme p_1 und p_2 können voneinander abweichen, obwohl die zugeordneten Programmkörper identisch sind.

Liste der Komponenten eines Programms

- Rufname p
- Programmkörper(p)
- $\text{Typ}(p)$: ein möglicher Wert kennzeichnet Dialogprogramme, der andere mögliche Wert kennzeichnet freilaufende Programme.
- $\text{PROG_SPRACHE}(p)$: gibt an, in welcher Programmiersprache das Quellprogramm zum Programmkörper(p) geschrieben ist.
- $\text{MAX_LÄNGE_KB_PROG}(p)$: gibt die maximale Länge des KB-Programmbereichs in PL über p an (vgl. Abschnitt 2.3.2.1.2). Wert aus dem zulässigen Zahlbereich; vgl. Abschnitt 2.1.2.
- ggf. sprachenabhängige Angaben über die Lage des Kommunikationsbereichs im Arbeitsspeicher von PL über p .
- (nur falls für p ein SPAB definiert wird)
 $\text{MAX_LÄNGE_SPAB}(p)$: gibt die maximale Länge des SPAB in PL über p an (vgl. Abschnitt 2.3.2.1.3). Wert aus dem zulässigen Zahlbereich; vgl. Abschnitt 2.1.2.

- ggf. sprachabhängige Angaben zum Aufbau des SPAB und zu seiner Lage im Arbeitsspeicher eines PL über p.

Datenstationsrufnamen

Es wird eine Menge M_DRN von Datenstationsrufnamen (das sind - syntaktisch gesehen - Rufnamen) beigebracht. Diese Menge M_DRN ist disjunkt zur Menge der eingeführten Programmrufrufenamen. Die Elemente von M_DRN heißen die eingeführten Datenstationsrufnamen des TAS.

Zu jedem $drn \in M_DRN$ wird eindeutig eine Typangabe mitgeteilt, nämlich

$DST_TYP(drn)$: dies ist eine Zeichenreihe der Länge 2 von folgender Art: Fz, Cz oder Ez, wobei z ein Zeichen aus dem Alphabet ist. Dies besagt, daß eine Datenstation, die drn zugeordnet ist, während der Zuordnung eine Typ-F (C; E)-Datenstation ist (zur Bedeutung vgl. Abschnitt 2.3.5.1).

Weiterhin wird zu jedem eingeführten Datenstationsrufnamen drn eindeutig eine Betriebsartangabe mitgeliefert.

$DST_ART(drn)$: dies ist eine syntaktisch nicht näher festgelegte Angabe zu den möglichen Betriebsrichtungen (Eingabe/Ausgabe) einer diesem Rufnamen zugeordneten Datenstation.

Die Betriebsartangabe ist entweder

- "ein- und ausgabefähig" oder
- "nur ausgabefähig".

Für die Sonderfunktionen der Eingabe (Eingabewerte für Funktionstasten) wird die Abbildung SC eines bestimmten Anfangsstücks von 01, 02, ..., 20 in die Menge 20, 21, ..., 39 festgelegt. (vgl. Abschnitt 2.3.5.4).

Datenstations- und Nachrichtenformate

Zu den Begriffen Datenstationsformat, Eingabenachrichtenformat, Ausgabenachrichtenformat vgl. Abschnitt 2.3.5.2.1.

Es werden Datenstationsformate

$DSTFORMAT = \{dstformat_1, \dots, dstformat_n\}$

definiert. Ihre während der Systemdefinition benötigten Bezeichnungen erscheinen nicht an der PL-Schnittstelle.

Außerdem werden Paare aus Formatkennzeichen und Nachrichtenformaten gebildet. Ein Formatkennzeichen nk bezeichnet eindeutig (nicht notwendigerweise eineindeutig) ein Nachrichtenformat: $NACHR\ FORMAT(nk)$.

Formatkennzeichen sind syntaktisch gesehen Zeichenreihen der Länge 8, für die gilt:

- das erste Zeichen ist ein Pluszeichen (+) oder ein Stern (*)
- dem ersten Zeichen folgt eine Zeichenreihe der Länge h , $1 \leq h \leq 7$, aus Buchstaben und Ziffernzeichen
- ist h kleiner als 7, so sind die Zeichen auf den Positionen $h+2$ bis 8 Leerzeichen.

Wenn zwei Formatkennzeichen der Gestalt $FormatKZ_1 = +zeichenreihe_1$ in der Gestalt $FormatKZ_2 = *zeichenreihe_2$ eingeführt werden, so muß immer $zeichenreihe_1 \neq Zeichenreihe_2$ sein. Diese Festlegung wird durch die Absicht begründet, daß

- ein Formatkennzeichen bereits ohne das erste Zeichen (+ oder *) eindeutig ein Nachrichtenformat bezeichnen soll.
- ein Nachrichtenformat entweder ausschließlich als 'Plusformat' oder ausschließlich als 'Sternformat' aufgerufen werden soll (zum Bedeutungsunterschied vgl. Abschnitt 2.3.5.2.1).

Die Formatkennzeichen der eingeführten Nachrichtenformate heißen eingeführte Formatkennzeichen des TAS.

Es werden Dreierformatgruppen festgelegt durch Tripel der Form

$[FormatKZ-a, dstformat, FormatKZ-e],$

wobei

- $FormatKZ-a$ ein eingeführtes Formatkennzeichen eines Ausgabenachrichtenformats ist.

- FormatKZ-e ein eingeführtes Formatkennzeichen eines Eingabem Nachrichtenformats ist.
- dstformat ein Element aus DSTFORMAT ist.

In diesen Tripeln darf kein Formatkennzeichen auf der ersten Position mehrfach auftreten, d.h. ein beteiligtes Formatkennzeichen für ein Ausgabenachrichtenformat kennzeichnet eine Dreierformatgruppe eindeutig.

Mit den Dreierformatgruppen werden weitergehende Zusammenhänge zwischen den beteiligten drei Formaten festgelegt, was insbesondere bedeutet, daß nicht beliebige Formate in Dreierformatgruppen zusammengefaßt werden können (vgl. Abschnitt 2.3.5.2.1).

Anmerkung: Ein Formatkennzeichen für ein Ausgabenachrichtenformat kennzeichnet eine Formatgruppe zwar eindeutig, kann also nicht in mehreren Formatgruppen erscheinen. Trotzdem kann dasselbe Ausgabenachrichtenformat in mehreren Formatgruppen auftreten; denn dasselbe Ausgabenachrichtenformat kann unter verschiedenen Formatkennzeichen durch die Abbildung NACHR_FORMAT erreicht werden.

Es werden Formatpaare festgelegt durch Paare der Art

[FormatKZ-a, dstformat]

mit analogen Erläuterungen und Festlegungen wie für die Dreierformatgruppen (anders ausgedrückt: die Formatpaare entsprechen rudimentären Dreierformatgruppen mit einer "leeren" dritten Komponente).

Systemparameter ohne Bezug zu Programmen oder Datenstationen

Die folgenden drei Parameter sind positive Zahlwerte aus dem zulässigen Zahlbereich (vgl. Abschnitt 2.1.2).

MAX_LÄNGE_SSB_SATZ: maximale Länge eines Satzinhalts im Lokalen oder im Globalen Sekundären Speicherbereich, gemessen in Byte (vgl. Abschnitte 2.3.3.1.3 und 2.3.4.2).

MAX_LÄNGE_NACHRICHTENTEIL: maximale Länge eines Nachrichtenteils in MPUT-, MGET-, FPUT-, FGET-Aufrufen (vgl. Abschnitte 2.3.3.1.2, 2.3.5 und 2.4.5), gemessen in Byte.

MAX_LÄNGE_SYSPROT_SATZ: maximale Länge des PL-Beitrags zu einem Satz im Systemprotokoll, gemessen in Byte (vgl. Abschnitt 2.3.5.5).

Weiter wird ein Vorbesetzungswert für den Parameter kcdf (Ausgabefunktion) bei Ausgaben an eine Datenstation (vgl. Abschnitt 2.3.5.2.1 f) benötigt.

Die Struktur der Datenstationsspeicherbereiche

Es ist die Länge L in Byte (positiv, aus dem zulässigen Zahlbereich; vgl. Abschnitt 2.1.2), einheitlich für alle Sätze in allen Datenstationsspeicherbereichen (DSB; vgl. Abschnitt 2.3.4.3) festzulegen, ferner eine Menge von Rufnamen, die Satzrufnamen der DSB-Sätze:

SRN_DSB = {srn₁, srn₂, ..., srn_n}. Eine systemweit eindeutige und zulässige Bezeichnung für einen DSB-Satz ist dann genau ein Paar, dessen erste Komponente ein eingeführter Datenstationsrufname, dessen zweite Komponente ein Element aus SRN_DSB ist (vgl. Abschnitte 2.3.4.1 und 2.3.4.3).

Anfangsbelegungen der globalen Speicher

Für die globalen Speicher (Globaler Sekundärer Speicherbereich (GSSB; vgl. Abschnitt 2.3.4.2), Datenstationsspeicherbereiche) können Anfangsbelegungen eingebracht werden. Dabei handelt es sich um eine gewisse Menge von Sätzen im GSSB sowie um Satzinhalte zu gewissen Satzrufnamen einiger Datenstationsspeicherbereiche. Diese Vorgaben müssen mit den festgelegten Parametern zur Struktur der globalen Speicherbereiche (MAX_LÄNGE_SSB_SATZ, Länge L und Rufnamen der DSB-Sätze, eingeführte Datenstationsrufnamen) konsistent sein.

Die FORMAT-Routine und die EXIT-Routine

Bei der Systemdefinition müssen auch die beiden installationspezifischen Programme

- FORMAT-Routine (vgl. Abschnitt 2.3.5.1)
 - EXIT-Routine (vgl. Abschnitt 2.3.6.1)
- eingebracht werden.

2.3 Dialogvorgänge und die DPL-Schnittstelle

2.3.1 Anforderung der Eröffnung und Grobstruktur von Dialogvorgängen

2.3.1.1 Anforderung der Eröffnung eines Dialogvorgangs

Ein Dialogvorgang wird immer durch Aktionen eines Benutzers an einer eingabefähigen Datenstation angefordert und danach vom TAS-Kern eröffnet. Dies setzt voraus, daß Eingabedaten von der fraglichen Datenstation den TAS-Kern erreichen.

Die die Vorgangseröffnung anfordernde Datenstation wird dann die Dialogdatenstation des Vorgangs, d.h. diejenige Datenstation, mit der der Dialogvorgang in einen Nachrichtenaustausch im strengen Wechselgespräch eintritt.

Anmerkung: Es soll nicht ausgeschlossen werden, daß durch besonderen Nachrichtenaustausch zwischen TAS-Kern und Benutzer eine andere Datenstation zur Dialogdatenstation des Dialogvorgangs gemacht wird. Eine solche Maßnahme wird vorzugsweise im Fehlerfall (Defekt an der Datenstation oder an der Transportverbindung) zu erwarten sein.

Die Eingabeaktionen, die ein Benutzer an einer geeigneten, aber aktuell keinem Dialogvorgang zugeordneten Datenstation vornehmen kann, um die Eröffnung eines Dialogvorgangs anzufordern, werden in ihrer Gesamtheit Prädialog genannt. Die Gestaltung des Prädialogs ist nicht Bestandteil der Norm. Aus dem Prädialog erhält der TAS-Kern folgende Datenobjekte:

- einen Programmrufrufnamen des ersten in dem Dialogvorgang zu startenden DP.
- eine Eingabenachricht, die zur Übernahme durch den ersten DPL vom TAS-Kern bereitgehalten wird (vgl. Abschnitt 2.3.5).
- nicht zwingend: eine Benutzeridentifikation in Gestalt einer Zeichenreihe der maximalen Länge 8 Zeichen.

Die Eingabenachricht kann vom ersten DPL

- als formatierte Nachricht gemäß einem aufgerufenen Eingabenachrichtenformat

- als unformatierte Nachricht

- durch Aufruf der FORMAT-Routine

übernommen werden. Die zu benutzenden TAS-Kernfunktionen und der Nachrichtenaustausch mit Datenstationen sind in Abschnitt 2.3.5 dargestellt.

Das Besondere der ersten Eingabenachricht in einem Dialogvorgang ist, daß die Eingabe an der Datenstation nicht mittels eines, mit der vorangegangenen Ausgabenachricht bereitgestellten Datenstationsformats geschehen kann (vgl. insb. Abschnitt 2.3.5.2.1/Formatgruppen). Der Benutzer kann möglicherweise im Prädialog ein (Eingabe-)Datenstationsformat aufrufen. Es ist auch möglich, daß der zuletzt an dieser Datenstation beendete Dialogvorgang sein letztes Datenstationsformat hinterläßt.

Bei unformatierter Eingabe wird eine Zeichenreihe eingegeben, deren erste 8 Zeichen den Programmrufrufnamen und deren Rest die unformatierte Eingabenachricht für den ersten PL darstellen.

Die Anforderung einer Vorgangseröffnung kann auch durch Betätigen einer Funktionstaste (vgl. Sonderfunktionen der Eingabe, Abschnitt 2.3.5.4) ausgedrückt und übermittelt werden.

Anmerkung: Es ist dann das Problem zu lösen, daß der erste PL die zum Prädialog passende Nachrichtenübernahme vornimmt, obwohl er vorher nur wenige Daten über seine aktuelle Auf-rufsituation erhalten kann. Dies geht grundsätzlich nur via KB-Kopf (vgl. weiter unten in diesem Abschnitt), wenn man den Weg über die globalen Speicher (Abschnitt 2.3.4) aus-schließt.

Bei Vorgangseröffnung wird der KB-Kopf (Kopf des Kommunikationsbereichs; bereits in Abschnitt 2.1.2 besprochen; weiteres in Abschnitt 2.3.2.1.2) für diesen Vorgang erstmals mit Werten be-legt. Der KB-Kopf enthält folgende Angaben:

- die Benutzeridentifikation - falls im Prädialog, bereitge-stellt, sonst 8 Leerzeichen
- der Programmrufrufname, unter dem das Programm des ersten PL im Prädialog bezeichnet wurde (auch "Vorgangsrufrufname")

- Datum und Zeit der Vorgangseröffnung

- Unterscheidungsmerkmal für Stellung des Programmlaufs im Vorgang (erster Programmlauf im Vorgang, nicht-erster Programmlauf im Vorgang)

Anmerkung: Diese Angabe kennzeichnet im übrigen den KB-Kopf eindeutig als denjenigen eines Dialogvorgangs. Bei einem freilaufenden Vorgang wird an der entsprechenden Stelle des KB-Kopfes ein ganz anderer Wert eingetragen (vgl. auch Abschnitt 2.4.1.1).

- der Programmrufrufname wie oben (der Wert auf dieser Position ist definiert als der Programmrufrufname zum aktuellen PL, bei dem es sich zur Vorgangseröffnung um den anschließend zu startenden PL handelt; vgl. auch Abschnitt 2.3.2.1.2)

- Startzeit des aktuellen PL (hier zunächst identisch mit der Zeit der Vorgangseröffnung)

- der Datenstationsrufrufname drn, den der TAS-Kern der Datenstation zugeordnet hat, von der aus die Vorgangseröffnung angefordert wurde. Dieser Datenstationsrufrufname, unter dem die Dialogdatenstation des Vorgangs erreicht wird, wird während des gesamten Vorgangs unter keinen Umständen verändert.

- der Typ des Datenstationsrufrufnamens DST_TYP(drn): vgl. Abschnitte 2.2 und auch 2.5.3.1.

- maximale Länge des KB-Programmbereichs (vgl. Abschnitt 2.3.2.1.2/INIT-Aufruf).

Zur genauen Lage und Gestaltung der Werte vgl. Tabelle 4 in Abschnitt 4.

Weiter wird der Lokale Sekundäre Speicherbereich (LSSB) für den Vorgang eingerichtet (vgl. Abschnitt 2.3.3.1.3), d.h. aber nicht, daß dieser Speicherbereich von diesem Zeitpunkt an notwendigerweise nennenswerte Betriebsmittel beansprucht, sondern nur, daß er mit den entsprechenden KA seitens der PL angesprochen werden kann.

Der Beginn des Dialogvorgangs ist gleichzeitig der Beginn des ersten Dialogschritts und damit der ersten Transaktion. Daraus ergibt sich die Notwendigkeit zu einigen Maßnahmen (vgl. Abschnitt 2.3.6).

2.3.1.2 Grobstruktur eines Dialogvorgangs (PEND)

Der Begriff des Dialogvorgangs - eingeführt in Abschnitt 0.3 - ist für die Wirkungsweise eines TAS-Kerns fundamental. Für die gesamte Dauer eines Dialogvorgangs bleibt der Datenstationsrufrufname, mit dem seitens der DPL des Vorgangs die Dialogdatenstation bezeichnet wird, unverändert.

Bei einem TAS-Kern heißt der zeitliche Ausschnitt aus einem Dialogvorgang, der zu einer vollständig abgehandelten Interaktion gehört - also vom Eintreffen einer Eingabenachricht bis zur Ausgabe der nächsten Nachricht an die Dialogdatenstation - ein Dialogschritt. Bereits in Abschnitt 0.3 wurde festgestellt, daß die auf Programme gestützten Aktivitäten innerhalb einer Interaktion, d.h. hier innerhalb eines Dialogschritts, eine Kette aus vollständigen PL bilden.

Grundsätzliches zu den PEND-Aufrufen

Der TAS-Kern erkennt spätestens beim PEND-Aufruf eines jeden PL, welche der folgenden Situationen vorliegt:

- Ende eines PL, aber nicht des Dialogschritts. Abschluß nur mit PEND-PA-Aufruf oder PEND-PR-Aufruf (PA aus Programmabschnitt, PR aus engl. program; die Unterscheidung ist historisch, die Aufrufe haben absolut gleiche Bedeutung).
- Ende eines PL und Ende des Dialogschritts, aber nicht Ende des Dialogvorgangs. Abschluß nur mit PEND-RE-Aufruf (RE aus engl. return).
- Ende eines PL und Ende des Dialogschritts und Ende des Dialogvorgangs. Abschluß nur mit PEND-FI (FI aus engl. final) oder - im vom PL erkannten Fehlerfall - mit PEND-ER (ER aus engl. error).

Jeder PEND-Aufruf, der nicht den Vorgang beendet, hat einen einzigen ergänzenden Parameter, nämlich den kern-Wert (rn für Rufname). Dieser Wert ist der Rufname desjenigen DP, das im nächsten PL dieses Vorgangs gestartet wird. (Falls der kern-Wert nicht der Rufname eines DP ist, so ergibt sich der Fehler 74%. Zu den Konsequenzen vgl. Abschnitt 2.3.6.1) Die vorgangsbeendenden PEND-Aufrufe haben überhaupt keinen ergänzenden Parameter.

Nur der letzte PL in einem Dialogschritt übergibt dem TAS-Kern die Nachricht, die bei Ende des Dialogschritts an der zugeordneten Dialogdatenstation auszugeben ist. Zum Nachrichtenaustausch mit der Dialogdatenstation vgl. Abschnitt 2.3.5.2.

Bei Beendigung eines Dialogvorgangs verschwinden die Programmläufe überdauernden Datenobjekte aus der Ausführungsumgebung des Vorgangs, insbesondere der KB und der LSSB, vgl. Abschnitt 2.3.7.

2.3.2 Programmlauflokale Speicher

2.3.2.1 Der Arbeitsspeicher (INIT)

2.3.2.1.1 Grundsätzliches

Der Begriff des Arbeitsspeichers eines PL sowie die in ihm realisierten Bytevektoren als Basis der Parameterübergabe wurden bereits in Abschnitt 2.1.2 eingeführt. Im übrigen ist die Art der Datenobjekte im Arbeitsspeicher von den verwendeten Programmiersprachen abhängig. Es gibt aber einige Datenobjekte im Arbeitsspeicher eines PL, von denen einige Eigenschaften an der PL-Schnittstelle bekannt sind und benützt werden. Diese Datenobjekte sind der bereits in den Abschnitten 2.1.2 und 2.3.1.1 behandelte Kommunikationsbereich (KB) und die sogenannten Primären Arbeitsbereiche (PAB).

Eigenschaften des KB-Programmbereichs und der PAB sind - bis auf ihre Länge in Byte - an der PL-Schnittstelle nicht bekannt. Angenommen wird, daß in ihnen andere Datenobjekte - insbesondere die Bytevektoren im Datenbereich eines KA - gespeichert werden können. Der Umstand, daß dem KB-Programmbereich und einem PAB eine Länge in Byte zukommt, besagt nicht, daß der KB-Programmbereich oder ein PAB ein Bytevektor im Sinne der PL-Schnittstelle sein muß. In einer geeigneten Programmiersprache kann er ein komplexes Datenobjekt sein; wie dessen Länge in Byte zu bestimmen ist, unterliegt dann einer programmiersprachenspezifischen Regelung. Während die an der PL-Schnittstelle relevanten Aspekte der Verwendung des KB bzw. der PAB in den Abschnitten 2.3.2.1.2 bzw. 2.3.2.1.3 dargestellt werden, ist hier eine wichtige generelle Grundregel über ihre Bedeutung für die Anordnung von Datenobjekten im Arbeitsspeicher anzugeben.

Regel (veränderliche Datenobjekte im Arbeitsspeicher)
Alle veränderlichen Datenobjekte eines PL, d.h. Datenobjekte, deren Wert möglicherweise während des PL durch Ausführung von Anweisungen des PL oder bei KA durch den TAS-Kern geändert werden, müssen im KB oder in PAB gespeichert sein.

Anmerkung: Der Hintergrund für diese Regel ist der eventuelle Einsatz abschnittsweise mehrfach verwendbarer Programmkörper. Bei Überholungen garantiert der TAS-Kern für einen PL nur den Erhalt der ihm bekannten Datenobjekte (KB und PAB) im Arbeitsspeicher des PL.

2.3.2.1.2 Der Kommunikationsbereich (INIT)

Der KB ist ein Bytevektor (Länge > 116), dessen Lage im Arbeitsspeicher dem TAS-Kern ab Beginn eines PL für dessen gesamte Dauer bekannt ist. Zur Gliederung des KB in seine drei Hauptteile - KB-Kopf, KB-Rückgabebereich, KB-Programmbereich - vgl. Abschnitt 2.1.2.

Regel 1 (KB-Initialisierung)

In Anweisungen eines PL darf vor dem INIT-Aufruf (vgl. Abschnitt 2.1.4) auf den KB weder lesend noch verändernd zugegriffen werden. Bei Verletzung dieser Vorschrift ist die Wirkung des PL - auch auf andere Teile des gesamten TAS - durch diese Norm nicht definiert.

Regel 2 (Nur Lesezugriffe zum KB-Kopf)

Auf den KB-Kopf darf in Anweisungen eines PL nur lesend zugegriffen werden. Bei Verletzung dieser Vorschrift ist die Wirkung des PL - auch auf andere Teile des gesamten TAS - durch diese Norm nicht definiert.

Die Länge des KB-Programmbereichs (LÄNGE_KB_PROG(PL)) wird durch den Bereichslängen-Parameter kobl des INIT-Aufrufs bestimmt, wobei dieser Wert mit dem bei der Systemdefinition für das Programm p des PL fixierten Wert MAX_LÄNGE_KB_PROG(p) verglichen wird. Es gilt:

$$\text{LÄNGE_KB_PROG(PL)} = \min(\text{kobl(INIT)}, \text{MAX_LÄNGE_KB_PROG(p)}).$$

Falls LÄNGE_KB_PROG(PL) = 0, so besteht der KB dieses PL nur aus dem KB-Kopf und dem KB-Rückgabebereich. Sonst ist der

$$\text{KB-Programmbereich} [1.. \text{LÄNGE_KB_PROG}] = \text{KB}[117..(116 + \text{LÄNGE_KB_PROG})].$$

Der gesamte KB existiert ausdehnungsmäßig und lagenmäßig während der gesamten Dauer eines PL im Arbeitsspeicher wie unmittelbar nach Ausführung des INIT-Aufrufs, d.h. insbesondere, daß der PL nicht Teile des KB-Programmbereichs "freigeben" oder "abmelden" kann.

2.3.2.1.3 Primäre Arbeitsbereiche (INIT, GTMM, FRMM)

Bei der Systemdefinition kann für ein Programm p ein Datenobjekt jeweils zur Realisierung im Arbeitsspeicher der PL über p definiert werden, der sog. Standard Primäre Arbeitsbereich (SPAB). Er ist disjunkt zum gesamten KB. Ein SPAB ist im Arbeitsspeicher jedes PL über p ab Beginn lagenmäßig und hinsichtlich seiner Zugriffseigenschaften fixiert. Trotzdem gilt die

Regel (SPAB-Initialisierung)

In Anweisungen eines PL darf vor dem INIT-Aufruf (vgl. Abschnitt 2.1.4) auf den SPAB weder lesend noch verändernd zugegriffen werden. Bei Verletzung dieser Vorschrift ist die Wirkung des PL - auch auf andere Teile des gesamten TAS - durch diese Norm nicht definiert.

Die Länge des SPAB (LÄNGE_SPAB(PL)) wird durch den Parameter PAB-Länge (kcpl) des INIT-Aufrufs bestimmt, wobei dieser Wert mit dem bei der Systemdefinition für das Programm p des PL fixierten Wert MAX_LÄNGE_SPAB(p) verglichen wird. Es gilt:

$$\text{LÄNGE_SPAB(PL)} = \min(\text{kcpl(INIT)}, \text{MAX_LÄNGE_SPAB(p)}).$$

Falls $\text{kcpl(INIT)} > \text{MAX_LÄNGE_SPAB(p)}$ gilt, so wird beim INIT-Aufruf als Statusanzeige kcsa eine Bemerkung (Wert 022) geliefert. Dies behindert aber nicht die Ausführung des Aufrufs gemäß obiger Längenformel.

Der SPAB erhält keine Vorbesetzung.

Falls $\text{kcpl(INIT)} > \text{MAX_LÄNGE_KB_PROG(p)}$ gilt, so wird beim INIT-Aufruf als Statusanzeige kcsa eine Bemerkung (Wert 012) geliefert. Dies behindert aber nicht die Ausführung des Aufrufs gemäß obiger Längenformel.

Der gesamte KB wird, soweit es die Länge des KB-Programmbereichs des in diesem Vorgang vorangehenden PL (falls es einen solchen gibt) gestattet, mit dem Inhalt des KB beim PEND-Aufruf des vorangehenden PL vorbesetzt (Der TAS-Kern darf für die Vorbesetzung des KB-Kopfes auch den Inhalt des KB-Kopfes beim Start des vorangehenden PL benutzen; dieser Inhalt muß zu diesem Zweck im TAS-Kern konserviert werden). Die Werte der Bytevektorkomponenten werden in diejenigen mit identischen Indexwerten übertragen. Ausgenommen sind generell zwei Werte im KB-Kopf:

- der Programmrufoffset (kcpr) zum aktuellen PL
- die Programmstartzeit (kcpz) des aktuellen PL.

Diese beiden Positionen sind nach dem INIT-Aufruf aktualisiert.

Nur beim Übergang vom ersten zum zweiten PL eines Dialogvorgangs gilt:

- der Wert der Einzelposition kcpv im KB-Kopf, die die Stellung des Programmlaufs im Vorgang betrifft, wird von 'F' (first) in 'N' (next) umgesetzt.

Zur Vorbesetzung des KB-Kopfes im ersten PL eines Vorgangs vgl. Abschnitt 2.3.1.1.

Wegen Regel 1 dieses Abschnitts und der Regel (KB-Rückgabebereich nach KA) aus Abschnitt 2.1.2 ist die Aussage über den KB-Transfer vom vorangehenden PL für den KB-Rückgabebereich bedeutungslos.

Falls die Länge des KB (Lalt) im vorangehenden PL kleiner war als die Länge des aktuellen KB (Lneu), so sind die Werte KB [Lalt+1..Lneu] nach dem INIT-Aufruf undefiniert. Falls es keinen vorangehenden PL gibt, ist Lalt = 116 anzusetzen.

Auf den Inhalt des KB-Kopfes wird nach Ausführung des INIT-Aufrufs während der gesamten Dauer des PL vom TAS-Kern höchstens lesend zugegriffen.

Der SPAB existiert während der gesamten Dauer eines PL im Arbeitsspeicher (abgesehen von Veränderungen seines Inhalts) wie unmittelbar nach Ausführung des INIT-Aufrufs, d.h. insbesondere, daß der PL den SPAB oder Teile davon nicht wie beispielsweise andere PAB freigeben kann, wie es nachfolgend in diesem Abschnitt beschrieben wird.

PL, die Hauptspeicheradressen manipulieren können, bei denen also die Hauptspeicherstruktur in die Arbeitsspeicherstruktur durchscheint, können die TAS-Kernfunktionen

GTMM (aus engl. get main memory) Gib Primären Arbeitsbereich
FRMM (aus engl. free main memory) Freigabe eines Primären Arbeitsbereichs

benutzen, um sich Primäre Arbeitsbereiche (PAB) zu beschaffen und auch wieder zurückzugeben.

Die wesentlichen ergänzenden Parameter eines GTMM-Aufrufs sind:

- ein Name b des geforderten PAB als Wert des Parameters kern.
Der Wert hat den Typ eines Rufnamens.
- die Länge des geforderten PAB in Byte als Wert des Parameters kodl.

Der Name b ist jeweils eine willkürliche Erfindung des PL und dient der Kommunikation an der konkreten Schnittstelle - insbesondere bei der Freigabe eines PAB.

Bei fehlerfreier Ausführung liefert der TAS-Kern eine Arbeitsspeicheradresse a im Parameter Bereichsadresse (kba). Im einfachsten Fall - d.h. bei einem byteadressierten Speicher - bedeutet dies, daß PL nunmehr die Adressen a, a+1, ... bis a + Länge -1 nutzen kann. (Bei nicht-byteadressierten Speichern ist das verfügbare Adressenintervall anders definiert, wobei der Längenanforderung in Byte sinngemäß - z.B. 4 Byte pro Zelle - Rechnung getragen wird.) Der gelieferte PAB ist disjunkt zu allen existierenden anderen PAB, dem SPAB und dem KB.

Beim GTMM kann es die Warnung

- kein Speicherbereich der angeforderten Länge vorhanden

geben (kcsa-Wert: 112). Ferner sind die Fehler

- angeforderte Länge außerhalb des zulässigen Zahlbereichs (43Z)
- unter dem Namen b gibt es einen PAB (geliefert und noch nicht wieder freigegeben). (44Z)

Der einzige ergänzende Parameter eines FRMM-Aufrufs ist der Name des freizugebenden PAB als Parameterwert von kern. Nach fehlerfreier Ausführung ist der zugeordnete Adressenbereich für PL nicht mehr nutzbar.

Als Fehler gibt es nur den Fall, daß

- kein PAB unter diesem Namen zugeordnet (kcsa-Wert 14Z)

ist.

2.3.2.1.4 Ergänzende Bemerkungen zum INIT-Aufruf (INIT)

Die wesentliche Wirkung des INIT-Aufrufs - außer, daß vor ihm kein anderer KA erfolgen kann - ist die Eröffnung des Zugriffs zum KB und ggf. zum SPAB durch die im PL ausgeführten Anweisungen.

Beim INIT-Aufruf können zwei noch nicht genannte schwerwiegende Fehler auftreten:

- eine der beiden Längenangaben liegt außerhalb des zulässigen Zahlbereichs (vgl. Abschnitt 2.1.2) (73Z)
- der KA kann wegen Systemfehler oder Generierfehler nicht durchgeführt werden (70Z).

Ein Generierfehler liegt beispielsweise vor, wenn im INIT-Aufruf der Zugang zu einem SPAB gefordert wird (kopl > 0), bei der Systemdefinition zu dem verwendeten Programm aber kein SPAB definiert wurde.

Wegen der Regeln zur KB- und zur SPAB-Initialisierung und wegen der Regel über veränderliche Datenobjekte im Arbeitsspeicher muß jeder PL in seinen Konstanten-Bereichen mindestens einen fertig ausgefüllten Parameterbereich für den INIT-Aufruf enthalten. Da der PL vor dem INIT-Aufruf auch auf keinerlei Eingabedaten zugreifen kann, können vorher höchstens Anweisungen deklarativen Charakters durchlaufen werden.

2.3.2.2 Der Lokale Sekundäre Speicherbereich (LSSB) als Programmmlauf-lokalen Speicher

Der einem Dialogvorgang zugeordnete LSSB (dargestellt in Abschnitt 2.3.3.1.3) kann und soll durchaus auch für die Zwecke eines einzigen PL genutzt werden, beispielsweise wenn nicht alle Programmmlauf-lokalen Zwischenergebnisse im Arbeitsspeicher gehalten werden können oder sollen.

Bei Gefahr von Namenskollisionen bei Satzruffamen zwischen verschiedenen PL eines Vorgangs, die den LSSB u.a. als lokalen Speicher benutzen, sollten die nicht weiterzugebenden Sätze vor PL-Ende aus dem LSSB gelöscht werden.

2.3.3 Systeminterne Datenübergabe von Programmmlauf zu Programmmlauf

Anmerkung: Die Datenübergabe auf dem Weg über globale Speicher (Abschnitt 2.3.4) wird hier nicht eingeordnet.

2.3.3.1 Vorgangsinterne Datenübergabe

2.3.3.1.1 Datenübergabe im Kommunikationsbereich

Die Datenübergabe von einem PL zum nachfolgenden durch Übergabe des gesamten KB oder eines Anfangsstücks des KB wird hier nur der Systematik halber nochmals erwähnt. Sie wurde in Abschnitt 2.3.2.1.2 behandelt bei der Frage nach der Vorbesetzung des KA durch den INIT-Aufruf.

2.3.3.1.2 Nachrichtenübergabe (MPUT, MGET)

Innerhalb eines Dialogvorgangs kann ein PL (hier PLvor genannt) dem unmittelbar nachfolgenden PL (hier PLnach genannt) durch Vermittlung des TAS-Kerns eine Nachricht übermitteln, falls zwischen den beiden PL kein Dialogschrittende liegt, d.h. falls nicht vom PLvor eine Nachricht an die Dialogdatenstation produziert wird. Nach Abschnitt 2.3.1.2 muß PLvor mit einem PEND-PA- oder einem PEND-PR-Aufruf beendet werden.

Eine Nachricht besteht aus einer einzigen Zeichenreihe, oder aus einer Folge von Zeichenreihen; die einzelnen Zeichenreihen,

die Elemente dieser Folge sind, heißen dann die Nachrichtenteile dieser Nachricht. Im erstgenannten Grenzfall sagt man: die Nachricht besteht nur aus einem Nachrichtenteil.

Die zu übermittelnde Nachricht wird in PLvor durch einen oder mehrere MPUT-Aufrufe (aus engl. put message) an den TAS-Kern übergeben (genau ein Nachrichtenteil pro Aufruf) und durch gleichviele MGET-Aufrufe (aus engl. get message) in PLnach in derselben Zerlegung in Nachrichtenteile dem TAS-Kern wieder abgenommen.

Ein MPUT-Aufruf hat eine der beiden Funktionsausprägungen NT (aus Nachrichtenteil) oder NE (aus Nachrichtenende). Der zweite unmittelbare Parameter PARAM2 wird benötigt. Die übrigen ergänzenden Parameter sind die Länge des Datenbereichs im Parameter kcnl (Länge des Nachrichtenteils in Byte) und ein Rufname im Parameter kcrn (der Programmurname des Programms von PLnach). Der Grenzfall kcnl=0 ist zulässig (es muß trotzdem ein zulässiger Wert für PARAM2 angegeben sein!). Der Wert von kcnl ist durch den Systemparameter (vgl. Abschnitt 2.2) MAX_LÄNGE NACHRICHTENTEIL nach oben begrenzt. Der einzige definierte Rückgabe-Parameter ist die Statusanzeige.

Sei MPUT-Folge = mput₁, mput₂, ... mput_r die Teilfolge aller KA von PLvor mit dem Funktionsmerkmal MPUT, wobei auch bei mput_r kein schwerwiegender Fehler auftreten möge. Dann sind die Schreibweisen wie

- Teilnachricht(mput_i): der Datenbereichsinhalt bei mput_i von der Länge kcnl(mput_i)
- kern(mput_i): der kern-Wert von mput_i
- Funktionsausprägung(mput_i): jeweils NT oder NE

unmittelbar verständlich.

Das relevante Anfangsstück der MPUT-Folge endet mit dem kleinsten ganzzahligen q , $1 \leq q < r$, für das gilt:

- (1) Funktionsausprägung(mput_q) = NE
oder (nicht ausschließend!)
- (2) kern(mput_q) ≠ kern(mput_{q+1})

Die KA $mput_1, \dots, mput_q$ heißen die relevanten MPUT-Aufrufe dieses DPL. Falls $q < r$, erhalten alle $mput_{q+1}, \dots, mput_r$ die Statusanzeige 41z: es werden Nachrichtenteile angeliefert, obwohl die Nachrichtenübernahme bereits abgeschlossen ist. Tritt der Sachverhalt nach Abbruchkriterium (2) auf - ev. gleichzeitig mit (1) - so wird ebenfalls der Fehler 41z gemeldet: hier bedeutet er einen Wechsel in der Folge der Rufnamen.

Der Rufname($mput_1$) = $korn(mp_1)$ heiße der Programmrufname der MPUT-Folge.

Die für MPUT-Aufrufe spezifischen schwerwiegenden Fehler sind

- die Längenangabe übersteigt die maximale Länge für Nachrichtenteile ($kcsa$ -Wert: 73z)
- der $korn$ -Wert ist nicht Rufname eines Dialogprogramms ($kcsa$ -Wert: 74z)

Falls der Programmrufname im PEND-Aufruf von PL_{vor} von dem Programmrufrufen seiner MPUT-Folge verschieden ist, wird der Vorgang beendet (vgl. Abschnitt 2.3.6) mit $kcsa$ -Wert 81z.

Ein MGET-Aufruf kennt keine Funktionsausprägung. Er benötigt ebenfalls den zweiten unmittelbaren Parameter PARAM2. Ergänzen der Aufrufparameter ($kcdl$) ist die Länge des Datenbereichs, der zur Aufnahme des Nachrichtenteils angeboten wird. Ergebnisparameter ist neben der Statusanzeige die Längenangabe ($kcla$) für den im Datenbereich gelieferten Bytevektor (= Nachrichtenteil).

Sei MGET-Folge = $mget_1, \dots, mget_s$ die Teilfolge aller KA von PL_{nach} mit dem Funktionsmerkmal MGET, wobei auch in $mget_s$ kein schwerwiegender Fehler auftreten möge. Jeder $mget_i$ ($1 \leq i \leq q$, q Länge des relevanten Anfangsstücks der MPUT-Folge in PL_{vor}) erhält als Ergebnis in seinem Datenbereich linksbündig den Nachrichtenteil($mput_i$) - $mput_i$ aus PL_{vor} -, sofern die Länge des Datenbereichs ausreicht. Falls der Nachrichtenteil kürzer ist als der Datenbereich, so ist der Inhalt des nicht belegten Teils des Datenbereichs undefiniert.

Andernfalls ($kcla(mget_i) > kcdl(mget_i)$) wird der gelieferte Nachrichtenteil rechts hinter dem $kcdl$ -ten Zeichen abgeschnitten. Als Statusanzeige wird eine Bemerkung ausgegeben ($kcsa$ -Wert = 01z). Falls $s > q$, so erhalten alle $mget_{q+1}, \dots, mget_s$ die Statusanzeige 10z: die Nachricht wurde bereits vollständig übernommen.

Gibt es in PL_{vor} gar keinen MPUT-Aufruf, so wirkt dies für MGET-Aufrufe in PL_{nach} so, als wäre in PL_{vor} genau ein MPUT-Aufruf mit der Datenbereichslänge 0 (Null) und dem $kcrn$ -Wert identisch mit dem Programmrufrufen aus dem PEND-Aufruf von PL_{vor} erfolgt.

Anmerkung: In den obigen Festlegungen ist u.a. ausgesagt, daß keine Nachrichtenteile von einem PL zu einem anderen als seinem unmittelbaren Nachfolge-PL übermittelt werden, d.h. beim Ende eines PL gehen für ihn bestimmte Nachrichtenteile, die er nicht gelesen hat ($s < q$), verloren.

2.3.3.1.3 Der Lokale Sekundäre Speicherbereich (LSSB) eines Dialogvorgangs (SPUT, SGET, SREL)

Zu jedem Dialogvorgang unterhält der TAS-Kern genau einen vorgangslkalen Speicher zur Speicherung von Sätzen unter beliebigen Rufnamen als Zugriffskennzeichen: den Lokalen Sekundären Speicherbereich (LSSB). Die Namen der Sätze heißen Satzrufnamen; syntaktisch sind sie Rufnamen (vgl. Abschnitt 2.2). Satz-inhalte sind Zeichenreihen, deren Länge durch einen systemglobalen Parameter - $MAX_LÄNGE_SSB_SATZ$ - begrenzt ist.

Zugriffe auf den LSSB aus den PL erfolgen ausschließlich durch KA. Jeder solche KA bezieht sich auf genau einen Satz. Es gibt keine implizite Satzauswahl; d.h. jeder Zugriff enthält explizit einen Satzrufnamen.

Es gibt eine TAS-Kernfunktion SPUT (aus engl. put secondary storage) zum Ablegen von Sätzen (drei Funktionsausprägungen: MS (aus engl. main storage), DL (aus engl. delay) und ES (aus engl. external storage), wobei MS und DL gleichbedeutend sind; Näheres weiter unten), eine TAS-Kernfunktion zum zerstörungsfreien Lesen eines Satzes (SGET-KP; SGET aus engl. get secondary storage und KP aus engl. keep) und eine zum zerstörenden Lesen (SGET-RL; RL aus engl. release), sowie eine TAS-Kernfunktion zum Löschen eines Satzes im LSSB (SREL-LB; SREL aus engl. release secondary storage und LB aus lokaler Bereich - zur Unterscheidung von SREL-GB, vgl. Abschnitt 2.3.4.2).

Der Satzinhalt wird grundsätzlich über den Datenbereich des KA übermittelt, d.h. SPUT und SGET-Aufrufe haben den unmittelbaren Parameter PARAM2. Die Länge des Datenbereichs steht in $kcdl$, der Satzrufname in $kcrn$.

Ein Überlauf des LSSB würde zu einem Fehler (kcsa-Wert = 40Z) führen: der TAS-Kern kann den Aufruf nicht ausführen.

2.3.3.2 Vorgangsgrenzen überschreitende Datenübergabe/ Eingangsdaten für freilaufende Vorgänge (FPUT, FGET)

In einem DPL (auch in einem FPL, vgl. Abschnitt 2.4.3) können Nachrichten für künftige FPL an den TAS-Kern übergeben werden. Dazu sind FPUT-Aufrufe (aus engl. put free message) zu benutzen, bei denen alle Details der Parameterversorgung mit denen bei MPUT-Aufrufen (vgl. Abschnitt 2.3.3.1.2) übereinstimmen, nur daß hier der Rufname (Wert von kern) der Name eines FP sein muß. Die Übergabe einer vollständigen Nachricht mittels eines oder mehrerer FPUT-Aufrufe stellt gleichzeitig die Anforderung beim TAS-Kern dar, einen FPL über dem genannten FP zu starten, was die (spätere) Eröffnung eines freilaufenden Vorgangs impliziert. Dieser FPL kann dann die fragliche Nachricht mittels einer Folge von FGET-Aufrufen (aus engl. get free message) beim TAS-Kern abholen, wobei die Details weitgehend mit denen beim MPUT/MGET-Zusammenspiel übereinstimmen. Über die weiteren notwendigen Voraussetzungen für die Eröffnung eines durch FPUT-Aufrufe angeforderten FPL (Verzögerung bis zum Dialogschritende) vgl. insbesondere Abschnitt 2.3.6.

FPUT-Aufrufe haben wie MPUT-Aufrufe eine der Funktionsausprägungen NT oder NE mit analoger Bedeutung, ferner einen Rufnamen als kern-Wert sowie den zweiten unmittelbaren Parameter PARAM2 und die dazugehörige Längenangabe im Parameter kcnl.

Anmerkung: Der Wert von kern in einem FPUT-Aufruf kann auch ein Datenstationsrufname sein. Dann liegt eine völlig andere Aufgabenstellung für den TAS-Kern vor; vgl. Abschnitt 2.3.5.

Für die genaue Abgrenzung einer zusammengehörigen Folge von FPUT-Aufrufen, die gerade eine Nachricht für ein zu startendes FPL vollständig übermitteln, ist es notwendig, von der Folge aller FPUT-Aufrufe im betrachteten PL auszugehen und diese zu zerlegen. In dieser Folge können - wie gesagt - neben FPUT-Aufrufen mit einem kern-Wert, der Programmrufname eines FP ist, auch FPUT-Aufrufe vorkommen, deren kern-Wert ein Datenstationsrufname ist.

Sei bei einem SPUT-Aufruf (gleichgültig mit welcher Funktionsausprägung) mit Satzrufnamen s der Datenbereich DB[1..kcdl]. Dann ist nach dem KA der Satzinhalt

$$s[j] = DB[j], \quad j = 1, \dots, \min(kcdl, MAX_LÄNGE_SSB_SATZ)$$

unter dem Namen s gespeichert. Falls kcdl > MAX_LÄNGE_SSB_SATZ ist, wird die Bemerkung 01Z als Statusanzeige geliefert.

Ein SPUT-MS-Aufruf übermittelt dem TAS-Kern die Vermutung des PL, daß noch in diesem PL auf den abgelegten Satz zugegriffen werden wird. Ein SPUT-ES-Aufruf übermittelt gerade die gegenteilige Vermutung. Falls in einem PL auf einen zuletzt mit SPUT-ES-Aufruf abgelegten Satz lesend zugegriffen wird, ergibt dies keinen Anlaß zu einer Statusanzeige ungleich "fehlerfreie Ausführung".

Bei einem SGET-Aufruf mit Satzrufnamen s und Datenbereich DB[1..kcdl] und dem unter s gespeicherten Satzinhalt s[1..h] ergibt sich

$$DB[j] = s[j], \quad j = 1, \dots, \min(kcdl, h).$$

Falls $h < kcdl$ ist, so sind die Werte von DB[h+1..kcdl] nach dem KA mit Leerzeichen belegt. Falls $h > kcdl$ ist, so zeigt die Statusanzeige fehlerfreie Ausführung an.

Bei SGET- und SREL-Aufrufen gibt es eine Warnung (kcsa-Wert = 14Z), falls kein Satz unter dem angegebenen Satzrufnamen im LSSB gespeichert ist.

Ein LSSB existiert genau für die Dauer des Dialogvorgangs, dem er zugeordnet ist. Er ist bei Vorgangsbeginn leer. Der Inhalt des LSSB wird ausschließlich durch Zugriffe der PL des Dialogvorgangs verändert, zu dem er gehört. Dies bedeutet insbesondere, daß bei Ende eines PL (ungleich Vorgangsende) der LSSB unverändert bleibt.

Ein SPUT-Aufruf überschreibt einen eventuell im LSSB existierenden Satz des angegebenen Namens. Ein SGET-Aufruf liefert genau dann den zuletzt unter demselben Satzrufnamen mittels SPUT-Aufruf abgelegten Satz, wenn seither weder ein SREL-Aufruf noch ein SGET_RL-Aufruf mit demselben Satzrufnamen erfolgt ist.

Ein SGET_RL-Aufruf wirkt wie ein SGET_KP-Aufruf unmittelbar gefolgt von einem SREL-Aufruf mit demselben Satzrufnamen - fehlerfreie Ausführung aller Aufrufe vorausgesetzt.

Sei $fput_1, fput_2, \dots, fput_r$ die Folge aller FPUT-Aufrufe im betrachteten PL in zeitlicher Anordnung. Es wird eine erste vollständige Teilfolge der FPUT-Folge gebildet nach der Regel für das relevante Anfangsstück einer MPUT-Folge (vgl. Abschnitt 2.3.3.1.2); sie sei $fput_1, \dots, fput_{q_1}$. Die nächste vollständige Teilfolge wird entsprechend als relevantes Anfangsstück der Restfolge $fput_{q_1+1}, \dots, fput_r$ gebildet u.s.w.f. bis $fput_1, fput_2, \dots, fput_r$ total in konsekutive, vollständige Teilfolgen beispielsweise mit den Endindizes $q_1 < q_2 < \dots < q_m = r$ zerlegt ist. Wird eine Teilfolge bei $fput_{q_i}$ nur wegen der Bedingung $kcrn(fput_{q_i}) \neq kcrn(fput_{q_{i+1}})$ abgeschlossen (d.h. daß gleichzeitig die Funktionsausprägung von $fput_{q_i}$ NT ist), so wird bei $fput_{q_{i+1}}$ die Bemerkung 04% (Rufname wechselt, ohne daß vorher ein FPUT-NE-Aufruf kam) als Statusanzeige geliefert. Zur i -ten Teilfolge $fput_{q_{i-1}+1}, \dots, fput_{q_i}$ (mit $q_0 = 0$ und $i = 1, \dots, m$) gehört als seine Zielangabe der $kcrn$ -Wert von $fput_{q_{i-1}+1}$, d.h. vom ersten FPUT-Aufruf in dieser Teilfolge. Diese Zielangabe kann ein Programmrufrufname oder ein Datenstationsrufrufname sein.

Die so definierten Teilfolgen der Folge aller FPUT-Aufrufe heißen die vollständigen FPUT-Folgen dieses PL.

Die vollständigen FPUT-Folgen, deren Zielangabe ein Datenstationsrufrufname ist, werden in Abschnitt 2.3.5.3 behandelt. Im Rest dieses Abschnitts wird unter einer vollständigen FPUT-Folge ausschließlich eine solche mit einem Programmrufrufnamen als Zielangabe verstanden.

Jede vollständige FPUT-Folge eines PL definiert genau eine Nachrichtenübergabe und gleichzeitig genau einen Startwunsch für einen FPL, nämlich über dem FP, dessen Rufname der Programmrufrufname der vollständigen FPUT-Folge ist.

In diesem FPL wirken die FGET-Aufrufe - sie seien $fget_1, fget_2, \dots, fget_s$ - hinsichtlich der vollständigen FPUT-Folge $fput_1, \dots, fput_{q_h}$ völlig analog wie MGET-Aufrufe bzgl. ihres relevanten Anfangsstücks der mput-Folge, d.h. $fget_j$ erhält den in $fput_{q_j}$ übergebenen Nachrichtenteil (Details über Längen etc. wie bei MPUT/MGET). Falls $s > h$ ist, erhalten die $fget_{h+1}, \dots, fget_s$ eine Statusanzeige 10% (Nachricht wurde bereits vollständig gelesen). Die Tatsache, daß an der DPL- bzw. an der FPL-Schnittstelle mit unterschiedlichen Funktionsmerkmalen MGET bzw. FGET identisch wirkende Funktionen bezeichnet werden, ist rein historisch zu verstehen.

Anmerkung: In einem PL können mehrere vollständige FPUT-Folgen mit demselben Programmrufrufnamen p vorkommen. Jede dieser FPUT-Folgen führt dann zu einem eigenen FPL über p .

2.3.4 Globale Speicher (SPUT, SGET, SREL, PTDA, GTDA)

2.3.4.1 Grundsätzliches zur Funktionsweise der globalen Speicher

Der TAS-Kern unterhält einige systemglobale, d.h. von vielen verschiedenen PL aus asynchron (d.h. ohne irgendeine explizite zeitliche Abstimmung zwischen den zugreifenden PL) zugreifbare Speicher. In ihnen werden Sätze unter Rufnamen als Zugriffskennzeichen gespeichert. Zugriffe auf globale Speicher erfolgen ausschließlich durch KA. Jeder solche KA bezieht sich auf genau einen Satz, dessen Satzrufname als Parameter des Aufrufs explizit anzugeben ist.

Es gibt zwei Arten von globalen Speichern: den Globalen Sekundären Speicherbereich, kurz: GSSB (es gibt genau einen im System) und die Datenstationsspeicherbereiche, kurz: DSB (von ihnen gibt es zu jedem eingeführten Datenstationsrufnamen im TAS genau einen). Für die DSB besteht eine generelle Zugriffsbeschränkung: aus einem Dialogvorgang, d.h. aus einem DPL, kann nur auf den DSB zugegriffen werden, der dem Datenstationsrufnamen (im KB-Kopf) der Dialogdatenstation dieses Dialogvorgangs zugeordnet ist; diese Beschränkung gilt für FPL nicht (vgl. Abschnitt 2.4.4).

Die Zugriffsarten zu den globalen Speichern sind Lesen, Schreiben und - nur im GSSB - auch Löschen eines Satzes.

Im folgenden wird nur über Zugriffe auf Sätze in irgendwelchen globalen Speichern gesprochen. Genau mit dem zeitlich ersten Zugriff, d.h. spätestens mit dem Ende des ersten durchgeführten KA dieser Art, auf einem Satz s aus einem Dialogschritt heraus wird dieser Satz zu Gunsten des betroffenen Dialogvorgangs gesperrt. Eine solche Satzsperrung kann keinesfalls zu Gunsten mehrerer Vorgänge gleichzeitig existieren. Alle Satzsperrungen zu Gunsten eines Dialogvorgangs werden bei einem Dialogschritt in diesem Vorgang aufgehoben. Satzsperrungen werden in völlig gleicher Weise auch zu Gunsten freilaufender Vorgänge gesetzt und aufgehoben, nur daß dem Zugriffsrahmen des Dialogschritts in freilaufenden Vorgängen der gesamte Vorgang entspricht (vgl. Abschnitt 2.4.4), d.h. die Sperren werden bei Vorgangsende aufgehoben.

Lesezugriff gibt es beim GSSB nicht. Die Länge der GSSB-Sätze ist durch denselben Systemparameter ($\text{MAX_LÄNGE_SSB_SATZ}$) begrenzt wie die der LSSB-Sätze. Auch die Behandlung von Längenunterschieden zwischen Datenbereich und Satz ist beim GSSB mit der beim LSSB gleich.

Der GSSB kann beim Systemstart bereits Sätze enthalten.

2.3.4.3 Der Datenstationsspeicherbereich des Dialogvorgangs (PTDA, GTDA)

Wie bereits in Abschnitt 2.3.4.1 festgestellt, kann aus einem Dialogvorgang nur auf einen DSB zugegriffen werden, nämlich denjenigen, der zu dem Datenstationsnamen gehört, der im KB-Kopf des Dialogvorgangs eingetragen und seinerseits der Dialogdatenstation des Vorgangs zugeordnet ist. Dadurch erübrigt sich in den entsprechenden KA eine explizite Adressierung des DSB. Es muß nur der Satzrufname angegeben werden.

Ein DSB kann n gleichlange Sätze der Länge L Byte unter den Satzrufnamen $\text{srn}_1, \text{srn}_2, \dots, \text{srn}_n$ speichern. n, L und die Namensmenge $\text{SRN_DSB} = \{\text{srn}_1, \text{srn}_2, \dots, \text{srn}_n\}$ sind Systemparameter und werden für alle DSB im System in gleicher Weise verwendet (vgl. Abschnitt 2.2).

In einem DSB gibt es Lese- und Schreibzugriffe mit Hilfe der Funktionsmerkmale GTDA (Lesen; aus engl. `get direct access`) und PTDA (Schreiben; aus engl. `put direct access`), beide ohne Funktionsausprägungen. Beide KA-Typen verlangen die Angabe eines Datenbereichs, dessen Länge im ergänzenden Parameter kcdl übermittelt wird. Der zweite ergänzende Parameter ist der Satzrufname, der als kcrn -Wert übermittelt wird.

Ein PTDA-Aufruf mit Satzrufnamen s und dem Datenbereichsinhalt $\text{DB}[1..kcdl]$ erzeugt den Satzinhalt $s[1..L]$, und zwar

$s[j] = \text{DB}[j]$, $j = 1, \dots, kcdl$, falls $kcdl \leq L$
und
 $s[j] = \text{Leerzeichen}$, $j = kcdl+1, \dots, L$, falls $kcdl < L$.

Falls $kcdl > L$ ist, wird $s[j] = \text{DB}[j]$, $j = 1, \dots, L$. Es wird die Statusanzeige 01z (der im Datenbereich gelieferte Bytevektor wurde rechts abgeschnitten) geliefert. In jedem Fall ist der Satzinhalt nach dem Schreibzugriff in voller Länge definiert.

Auf einen Satz, der zu Gunsten eines anderen Vorgangs gesperrt ist, kann kein anderer Vorgang einen Zugriff ausführen; er kann zwar einen entsprechenden KA beginnen, aber dieser kann nicht erfolgreich zu Ende geführt werden, solange eine Sperre auf dem betroffenen Satz zu Gunsten eines anderen Vorgangs besteht.

Anmerkung: Die Satzsperrung der globalen Speicher ist nicht differenziert, d.h. insbesondere, daß nicht zwischen Lesesperre und Schreibsperrung unterschieden wird.

Falls ein KA bzgl. eines globalen Speichers mit der Statusanzeige 40z (TAS-Kern kann den Zugriff nicht ausführen) beendet wird, so hat sich durch diesen KA bzgl. des betroffenen Satzes für den aufrufenden Vorgang nichts geändert – weder beim Sperrzustand noch beim Satzinhalt. Für die Statusanzeige 40z kann es verschiedene Gründe geben wie z.B. Überlauf des GSSB (nicht bei den DSB, vgl. Abschnitt 2.3.4.3). Aus dem bisher Gesagten folgt, daß sich vor einem Satz s Wartemengen oder Warteschlangen von Vorgängen bilden können, die auf s zugreifen wollen, aber keine Sperre bzgl. s zu ihren Gunsten haben. Der TAS-Kern muß dann, sobald auf s keine Sperre mehr besteht, in angemessener Frist zu Gunsten genau eines der zugriffswilligen Vorgänge eine Sperre auf s einrichten.

Anmerkung: Die Norm macht keine Aussage über die Reihenfolge, in der aus einer Menge wartender Vorgänge der Nächste ausgewählt wird. Die Norm macht auch keine Vorschriften über das Erkennen und ggf. über die Behandlung von Verklemmungen.

2.3.4.2 Der Globale Sekundäre Speicherbereich (SPUT, SGET, SREL)

Zugriffe auf den GSSB erfolgen mit den TAS-Kernfunktionen SGET-GB (Satz zerstörungsfrei lesen; GB aus globaler Bereich), SPUT-GB (Satz schreiben), SREL-GB (Satz löschen). Als Satzrufnamen sind beliebige Rufnamen zugelassen. Abgesehen von den Zugriffseigenschaften des GSSB als globaler Speicher gemäß Abschnitt 2.3.4.1 wirken die KA wie die entsprechenden Zugriffe auf einen LSSB (vgl. Abschnitt 2.3.3.1.3). Die Parameterversorgung ist mit der der entsprechenden LSSB-Aufrufe identisch – abgesehen eben von den Funktionsausprägungen. Eine Unterscheidung mehrerer SPUT-Varianten gibt es beim GSSB nicht. Ein SGET-GB-Aufruf entspricht einem SGET-KP-Aufruf. Einen zerstörenden

Ein GTDA-Aufruf mit Satzrufnamen s und dem Datenbereich DB[1..kcdl] führt zur Belegung

$DB[j] = s[j], j = 1, \dots, \min(L, kcdl).$

Es gibt keine besondere Statusanzeige für den Fall $kcdl < L$. Falls dem GTDA-Aufruf (seit Systemstart) noch kein PTDA-Aufruf mit demselben Satzrufnamen vorangegangen ist, so ist der Wert von $s[j]$ ein Leerzeichen, $j = 1, \dots, L$; einige der Sätze eines DSB können abweichend davon ab Systemstart aber auch nicht-triviale Satzinhalte haben, wie z.B. Zugangsregulierungen für die jeweilige Datenstation, die dann durch einen solchen GTDA-Aufruf gelesen werden.

Beide Aufrufe kennen die Statusanzeige "kern-Wert gehört nicht zu der für DSB festgelegten Menge von Satzrufnamen". Dies ist ein Fehler mit kosa-Wert = 44Z.

Eine Lösche-Funktion - analog SREL bei LSSB und GSSB - gibt es für DSB nicht.

2.3.5 Nachrichtenaustausch mit Datenstationen / Systemprotokoll

2.3.5.1 Datenstationstypen und Aufrufarten (MGET, MPUT, FPUT)

Der TAS-Kern unterstützt den Nachrichtenaustausch zwischen Vorgängen und Datenstationen. Dabei werden seitens der PL die KA mit den Funktionsmerkmalen MPUT, MGET und FPUT verwendet (Verwendung von FGET-Aufrufen in freilaufenden Vorgängen vgl. Abschnitt 2.4.5). Eingabenachrichten werden beim TAS-Kern zwischengespeichert und seitens der PL durch MGET-Aufrufe abgenommen. Ausgabenachrichten werden dem TAS-Kern durch MPUT- oder FPUT-Aufrufe übergeben. Die gerade genannten Funktionsmerkmale werden auch bei KA zur systeminternen Datenübergabe verwendet (vgl. insbesondere Abschnitt 2.3.3). In diesem Hauptabschnitt Abschnitt 2.3.5 wird nur der die Systemgrenzen überschreitende Nachrichtenaustausch mit Datenstationen behandelt, was dann im Einzelfall nicht ständig wieder gesagt wird.

Bei den Einzelaktionen zu einem Nachrichtenaustausch an der PL-Schnittstelle sind zwei Arten von Bestimmungsgrößen von besonderer Bedeutung:

- der Typ des benutzten Datenstationsrufnamens (F, C oder E)
- die Art des KA (formatgebunden, formatfrei, FORMAT-Routine)

Beide Angaben finden sich - direkt oder indirekt - in den Parametern der KA.

Da sich der Typ und die Art eines Datenstationsrufnamens während der Zuordnung zu einer Datenstation auf diese übertragen (vgl. Abschnitt 2.1), wird im folgenden die Bedeutung des Typs unter Bezugnahme auf die Datenstation erläutert. Man unterscheidet

- Typ-F-Datenstationen, auch Formatdatenstationen genannt. Für ihren Betrieb gibt es vordefinierte, bei der Systemdefinition hinterlegte, benannte Formate und zwar Nachrichten- und Datenstationsformate. Näheres in Abschnitt 2.3.5.2.
- Typ-C-Datenstationen. Ihr Betrieb wird vom unterliegenden Wirtssystem (DC-System) soweit unterstützt, daß die Vorgänge mit ihnen unformatierte Nachrichten austauschen können.
- Typ-E-Datenstationen. Bei ihrem Betrieb wird die Nachrichtenumsetzung auf dem Weg zwischen Datenstation und PL von einer FORMAT-Routine - mit einer in der Norm nicht näher festgelegten Leistung - vorgenommen. Bei einem KA zum Nachrichtenaustausch wird, falls der benutzte Datenstationsrufname vom Typ E ist, die FORMAT-Routine automatisch - d.h. ohne besondere Maßnahmen des PL - aufgerufen und mit allen Daten des KA versorgt. Sie liefert bzw. entnimmt dann den anstehenden Nachrichtenanteil in den bzw. aus dem Datenbereich des KA. Die FORMAT-Routine - die auch bei der Systemdefinition in das System eingebracht wird - hat eine nicht genormte Schnittstelle zum TAS-Kern. Die FORMAT-Routine kann, obwohl sie im weiteren Sinn in den Bereich der Anwendungsprogramme gehört, die genormten Dienste der PL-Schnittstelle in Form von KA nicht benutzen.
- Bei den KA zum Nachrichtenaustausch unterscheidet man drei den Datenstationstypen entsprechende Arten, die am ersten Zeichen des Formatkennzeichens (Parameter knck) erkennbar sind:
 - Formatgebundener Aufruf; Austausch formatierter Nachrichten(-teile). Es wird auf ein in der Systemdefinition eingeführtes Nachrichtenformat explizit Bezug genommen.
 - Formatfreier Aufruf; Austausch unformatierter Nachrichten(-teile). Aus der Sicht der PL-Schnittstelle sind die Nachrichten(-teile) nicht weiter strukturierte Zeichenreihen.

- (Quasi) Expliziter Aufruf der FORMAT-Routine. (Die FORMAT-Routine wurde oben bei der Einführung des Datenstationstyps E erläutert).

Die Aufrufarten und die Datenstationstypen sind als Elemente eines KA nur in einem einzigen Fall unverträglich, nämlich in der Kombination Typ-C-Datenstation mit formatgebundenem Aufruf.

2.3.5.2 Nachrichtenaustausch mit der Dialogdatenstation des Vorgangs (MGET, MPUT, PEND-RE, PEND-FI)

2.3.5.2.1 Austausch formatierter Nachrichten

a) Einleitung

Eine besondere Unterstützung erfährt der Nachrichtenaustausch mit den Formatdatenstationen wie es beispielsweise Bildschirmdatenstationen aber auch reine Ausgabedatenstationen wie Terminaldrucker sein können.

Es werden Datenstationsformate sowie Nachrichtenformate benutzt, wobei noch zwischen Nachrichtenformaten für die Eingabe und Nachrichtenformaten für die Ausgabe unterschieden wird. Diese Formate werden bei der Systemdefinition festgelegt und benannt und sind nach dem Systemstart weder veränderbar noch umbenennbar. Auf diese Nachrichtenformate kann direkt von den PL aus durch Nennung der Formatkennzeichen Bezug genommen werden.

Die Formate werden stets in gewissen Gruppierungen (Formatgruppen) gebraucht. Über das Kennzeichen eines Nachrichtenformats für die Ausgabe findet der TAS-Kern das eindeutig zugehörige Datenstationsformat. Falls die Datenstation als Ausgabedatenstation angesprochen wird (DST_ART (drn) = "nur ausgabefähig"; vgl. Abschnitt 2.2), genügt die Paarung Ausgabedatenstationenformat / Datenstationsformat. Ist die Datenstation die Dialogdatenstation eines Vorgangs, so gehört zu dem benannten Ausgabedatenstationsformat auch noch eindeutig ein Eingabenachrichtenformat, nämlich zur Gestaltung der im Dialog folgenden nächsten Eingabenachricht (vgl. Dreierformatgruppen und Formatpaare, Abschnitt 2.2).

Wichtige Bestandteile von Nachrichten sind Zeichenreihen über dem Alphabet nach Abschnitt 2.1.2.

b) Die Detailstruktur der Formate und ihre Beziehungen in einer Formatgruppe

Es werden die drei Formatarten - Datenstationsformate, Ausgabeformatenformate, Eingabenachrichtenformate - und ihre Zusammenhänge in einer Dreierformatgruppe betrachtet. Danach wird ihre Nutzung beim Nachrichtenaustausch zwischen einem Dialogvorgang und einem Benutzer an einer Dialogdatenstation dargelegt. Die Merkmale eines Formatpaares ergeben sich durch Weglassen einiger Merkmale einer Dreierformatgruppe.

b.1) Begriff des Datenstationsformats

An einer Formatdatenstation können Zeichenreihen vorzugsweise auf einer rechteckigen Fläche, dem Darstellungsbereich, dargestellt werden. Für eine Klasse gleicher oder hinreichend ähnlicher Datenstationen werden Datenstationsformate definiert, indem pro Format vor allem eine gewisse Anzahl von nicht-überlappenden Teilbereichen des Darstellungsbereichs (die Felder des Formats) ausgewiesen werden, naheliegenderweise durch ihre geometrische Lage im Darstellungsbereich. Jedes Feld kann eine Zeichenreihe der Länge L aufnehmen, wobei diese Längen für die verschiedenen Felder eines Formats verschieden sein können.

Die durch ein Datenstationsformat dst-format definierten m Felder im Darstellungsbereich (in vereinfachender Ausdrucksweise: die Felder von dst-format) werden unterschieden in Ausgabefelder (A-Felder) und Eingabefelder (E-Felder), wobei es i.allg. Felder gibt, die sowohl A- als auch E-Felder sind (AE-Felder). Ein Feld, das nur A-Feld bzw. E-Feld ist, heißt reines A-Feld bzw. reines E-Feld.

Zu jedem reinen E-Feld enthält ein Datenstationsformat eine fest vorgegebene Zeichenreihe entsprechender Länge als Vorsetzungswert.

Anmerkung: Diese Vorsetzungswerte der reinen E-Felder können im Grenzfall völlig nichtssagend sein; sie können beispielsweise für alle Datenstationsformate grundsätzlich aus Zwischenraumzeichenreihen bestehen.

Weiter gehören zu einem Eingabenachrichtenformat drei Initialisierungswerte: INIT1, INIT2, INIT3. Jeder ist ein Zeichen aus dem Alphabet. Ihre Bedeutung wird weiter unten beschrieben.

Als Hilfestellung für den Benutzer werden mit einem Datenstationsformat üblicherweise auch gewisse konstante Belegungen (Zeichenreihen oder Graphikelemente) definiert, die mit den oben genannten Feldern nicht überlappen. Diese Teile eines Datenstationsformats spielen im folgenden überhaupt keine Rolle mehr, da sie beim Nachrichtenaustausch keinerlei Funktion haben.

b.2) Attribute der Felder eines Datenstationsformats

Jedem Feld eines Datenstationsformats ist eine Kombination von Attributwerten zugeordnet. In einer solchen Attributwertkombination ist jeweils genau ein Wert zu jeder der im folgenden beschriebenen Attributkategorien enthalten. Diese Gruppe der Attributkategorien mit ihren Werten gibt den aktuellen Stand wieder (1983/84). Sie wird in geeigneten Zeitabständen der gerätetechnischen Entwicklung angepaßt. Die Darstellung der Werte durch Zeichenreihen - sofern festgelegt - ist in Klammern angegeben.

- Angabe zum Zeichensatz. Werte:
 - alphanumerisch ('ALPH'). Zeichen aus dem genannten Alphabet sind zur Bildung des Feldinhalts zugelassen.
 - numerisch ('NUME'). Nur die Ziffernzeichen sowie die Zeichen *+/-/./, und Schrägstrich sind zur Bildung des Feldinhalts zugelassen.
- Angabe zur Veränderbarkeit durch den Bediener. Werte:
 - ungeschützt ('UNPR'; aus unprotected). E-Feld, dessen Inhalt der Bediener verändern kann.
 - geschützt ('PROT'; aus protected). E-Feld, dessen Inhalt der Bediener nicht verändern kann.
- Angabe zur Darstellungsweise des Feldinhalts. Werte:
 - normal leuchtend ('NINT'; aus normal intensity). Der Feldinhalt hat normale Helligkeit.
 - dunkel ('DINT'; aus dormant). Der Feldinhalt ist für den Bediener nicht lesbar.
 - hell leuchtend ('HINT'; aus high intensity). Der Feldinhalt hat maximale Helligkeit.
 - kursiv ('ITAL'; aus italics). Die Zeichen im Feld sind kurziv geschrieben.
 - blinkend ('SIGN'; aus signalling). Der Feldinhalt blinkt.

- Angabe zur Auswählbarkeit mittels Lichtgriffel. Werte:
 - auswählbar ('DETE'; aus detectable). Das Feld, das notwendigerweise auch E-Feld ist, ist mittels Lichtgriffel auswählbar.
 - nicht auswählbar. Negation der Bedeutung des ersten Wertes.
- Angabe zum Eingabezwang von auch nicht vom Bediener veränderten Feldinhalten. Werte:
 - a priori verändert ('PREM'; aus premodified). Der Feldinhalt wird a priori als vom Bediener verändert betrachtet und bei der Eingabe entsprechend behandelt.
 - nicht a priori verändert. Negation der Bedeutung des ersten Wertes.
- Angabe zur Schreibmarke (Cursor). Werte:
 - gesetzt. Die Schreibmarke steht auf der ersten Zeichenposition dieses Feldes.
 - nicht gesetzt. In dem Feld steht keine Schreibmarke.

Anmerkung: Da üblicherweise nur eine Schreibmarke existiert, ist es nicht sinnvoll, bei mehreren Feldern die Schreibmarke gesetzt zu haben.

Nicht alle Attributwertkombinationen sind zulässig. Die Werteverbindung 'auswählbar' und 'a priori verändert' ist unzulässig, ebenso die Verbindung 'auswählbar' mit 'dunkel' (vgl. Tabelle 8 in Abschnitt 4).

b.3) Nachrichtenformate

Im folgenden wird von den drei Formaten einer Formatgruppe gesprochen. In diesem Sinne ist dann der bestimmte Artikel zu verstehen: das Ausgabenrichtenformat (der Gruppe), das Datenstationsformat (der Gruppe).

Ein Ausgabenrichtenformat besteht aus einer Folge von Feldbeschreibungen, die ihrerseits hier nur Feldlängenangaben, also natürliche Zahlen (ungleich Null), sind. Dabei ist die Reihenfolge der angegebenen Feldlängen wesentlich. Man kann sagen: ein Ausgabenrichtenformat gibt p Felder vor, sukzessive mit den Längen $\text{feldlänge}_1, \text{feldlänge}_2, \dots, \text{feldlänge}_p$.

Die Felder eines Ausgabenrichtenformats sind den A-Feldern des Datenstationsformats paarweise eindeutig zugeordnet, wo-

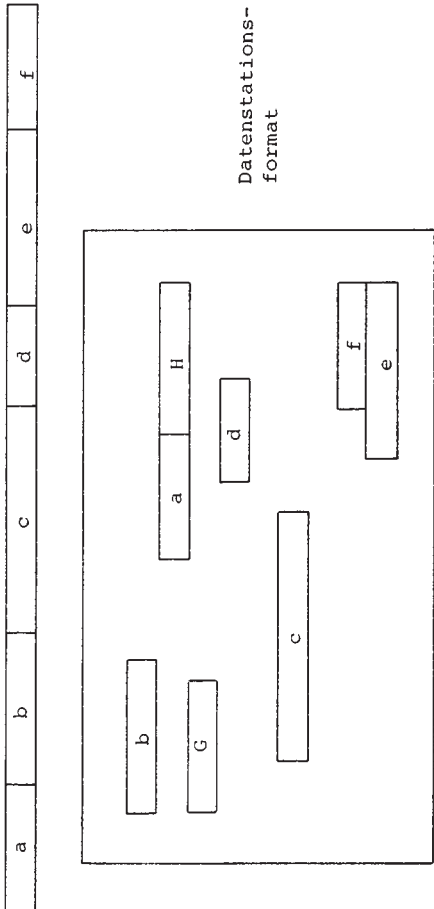
raus folgt, daß die Anzahl der Felder des Ausgabenachrichtenformats und die Anzahl der A-Felder des zugeordneten Datenstationsformats gleich sein müssen. Ferner gilt: die Länge eines Feldes des Ausgabenrichtenformats ist identisch mit der Länge des ihm zugeordneten A-Feldes des Datenstationsformats.

Ein Eingabenachrichtenformat beschreibt eine Folge von Feldern. Die Feldbeschreibungen, die eine Folge bilden, sind in zwei sich ausschließende Arten aufgeteilt: Beschreibungen echter E-Felder und Beschreibungen von Literalfeldern. Jede Feldbeschreibung - egal welcher Art - enthält eine Feldlängenangabe. Zu jedem Literalfeld gehört eine fest vorgegebene Zeichenreihe (Literal), deren Länge gleich der für dieses Feld angegebenen Feldlänge ist.

Die echten E-Felder des Eingabenachrichtenformats sind den E-Feldern des Datenstationsformats paarweise eineindeutig zugeordnet, woraus folgt, daß die Anzahl der echten E-Felder des Eingabenachrichtenformats und die Anzahl der E-Felder des Datenstationsformats gleich sein müssen. Ferner gilt: die Feldlänge aus einer Feldbeschreibung eines echten E-Feldes ist identisch mit der Länge des ihm zugeordneten E-Feldes des Datenstationsformats. Für die echten Eingabefelder, die reinen Eingabefeldern des Datenstationsformats zugeordnet sind, sind im Eingabenachrichtenformat Vorbesetzungswerte definiert.

Es folgt die geometrische Veranschaulichung von Zusammenhängen zwischen den verschiedenen Formaten an Hand eines einfachen Beispiels. Die Benennungen der Felder dienen nur dazu, die Zuordnungen auszudrücken.

Ausgabenachrichtenformat



Eingabenachrichtenformat

XXXXX	d	YYYYY	G	b	H	f
-------	---	-------	---	---	---	---

- XXXXX, YYYYY Literale (die Felder sind Literalfelder)
- a, c, e reine Ausgabefelder
- b, d, f AE-Felder
- G, H reine Eingabefelder
- b, d, f, G, H echte Eingabefelder

Bild 1. Beispiel für eine Formatgruppe

c) Dynamische Abwicklung von Ausgabe und Eingabe

Ein vollständiger Nachrichtenaustausch zwischen einem Dialogvorgang und dem korrespondierenden Benutzer an der zugeordneten Dialogdatenstation dñst beginnt mit der Ausgabe einer Ausgabemessage a-nachricht, wodurch ein bestimmter Bildschirminhalt a-bildschirminhalt entsteht. Diesen wird dann der Benutzer im Regelfall zu einem Bildschirminhalt e-bildschirminhalt verändern, aus dem unter Beachtung der aufgerufenen Formatgruppe (siehe heiße: a-format, dñst-format, e-format) eine Eingabemessage e-nachricht entsteht, die im Dialogvorgang verarbeitet werden kann.

Es werden zunächst nur einteilige Nachrichten betrachtet. Zu mehrteiligen Nachrichten vgl. Abschnitt 2.3.5.2.1 e).

c.1) Definition des Begriffs Bildschirminhalt

Ein Bildschirminhalt zu einem Datenstationsformat dñst-format besteht aus Zeichenreihen passender Art und Länge, die den Feldern von dñst-format zugeordnet sind. Außerdem gehört zu jedem Feld eine Attributwertkombination, die außer den Werten nach den bereits mit dem Datenstationsformat eingeführten Attributkategorien bei den E-Feldern noch zwei weitere Werte umfaßt aus den folgenden Attributkategorien:

- Veränderungszustand. Werte:
 - verändert. Der Feldinhalt wurde seit der letzten Ausgabe vom Benutzer verändert.
 - nicht verändert. Negation der Bedeutung des ersten Wertes.
- Auswahlzustand. (Nur für die Felder, die im Datenstationsformat den Attributwert 'auswählbar' haben) Werte:
 - ausgewählt. Das Feld wurde seit der letzten Ausgabe vom Benutzer mittels Lichtgriffel ausgewählt.
 - nicht ausgewählt. Negation der Bedeutung des ersten Wertes.

Diese beiden Attributwerte pro Feld sind beim Erzeugen einer Eingabemessage aus einem Bildschirminhalt bedeutungsvoll (vgl. den Abschnitt 2.3.5.2.1 c.4)).

Hinweis: Die Attributwertkombination eines Feldes eines Bildschirminhalts gemäß einem Datenstationsformat dñst-format muß für die im Datenstationsformat definierten Attributkategorien -

keineswegs mit den in dñst-format festgelegten Attributwerten übereinstimmen. Vgl. Wirkung einer Ausgabemessage, die im unmittelbar folgenden Teilabschnitt erläutert wird.

c.2) Erzeugung eines Bildschirminhalts durch Ausgabe einer Nachricht

Es gibt zwei grundsätzlich verschiedene Arten von Ausgabemessages: solche ohne und solche mit Attributwertangaben.

Eine Ausgabemessage a-nachricht ohne Attributwertangaben zu einem Ausgabemessagesformat a-format = (feldlänge₁, feldlänge₂, ..., feldlänge_p) besteht aus einer Folge von p Zeichenreihen z₁, z₂, ..., z_p, wobei die Länge von z_i gleich feldlänge_i ist, i = 1, ..., p.

Wird a-nachricht unter Aufruf des Ausgabeformats der Formatgruppe a-format, dñst-format, e-format ausgegeben, so entsteht folgender Bildschirminhalt a-bildschirminhalt (eine Einteilung der beteiligten Nachrichten bzw. des Bildschirms gemäß den aufgerufenen Formaten wird im folgenden vorausgesetzt):

- (1) die Ausgabefelder des Bildschirms werden mit den zugeordneten Zeichenreihen aus a-nachricht belegt.

Anmerkung: Eine Zeichenreihe, die in ein Feld mit dem Attributwert 'numerisch' übertragen wird, darf nur numerische Zeichen enthalten.

- (2) die reinen Eingabefelder des Bildschirms werden mit den dafür im Eingabemessagesformat definierten Vorsetzungswerten belegt.

- (3) bzgl. der im Datenstationsformat definierten Attributkategorien enthalten die Attributwertkombinationen der Felder die in dñst-format festgelegten Werte.

- (4a) alle dafür in Frage kommenden Felder haben den Attributwert 'nicht ausgewählt'.

- (4b) alle E-Felder haben den Attributwert 'nicht verändert' außer denjenigen E-Feldern, die gemäß (3) den Attributwert 'a priori verändert' erhalten haben. Diese erhalten den Attributwert 'verändert'.

Eine Ausgabenachricht mit Attributwertangaben zu dem a-format = (feldlänge₁, ..., feldlänge_p) ist eine Folge von p Paaren der Art
[Attributwertkombination, Zeichenreihe].

Die Zeichenreihe aus dem i-ten Paar hat die Länge feldlänge_i, i = 1, ..., p. Die zulässigen Attributwertkombinationen sind genau die bei der Definition des Begriffs Datenstationsformat eingeführten. Zusätzlich gibt es hier die leere Attributwertkombination, allerdings in zwei Varianten: mit und ohne gesetzte Schreibmarke.

Die Attributwertkombinationen in einer Ausgabenachricht können nicht ganz unabhängig von der entsprechenden Attributwertkombination im Datenstationsformat gewählt werden: nicht-auswählbare Felder im Sinne des Datenstationsformats können so nicht zu auswählbaren Feldern gemacht werden.

Die Aussagen über die Erzeugung eines Bildschirmformats durch Ausgabe einer Ausgabenachricht mit Attributwertangaben ergeben sich aus den obigen Aussagen für Ausgabenachrichten ohne Attributwertangaben durch folgende Abänderungen.

Statt (3) gilt hier:

(3a) bzgl. der im Datenstationsformat definierten Attributkategorien erhält ein A-Feld des Bildschirmformats die jeweils durch die Formatzusammenhänge zugeordnete Attributwertkombination aus der Ausgabenachricht, sofern diese nicht die leere Attributwertkombination ist; in diesem Fall erhält das Feld die entsprechende Attributwertkombination aus dst-format.

Eine leere Attributwertkombination mit Schreibmarke in der Ausgabenachricht führt beim entsprechenden Feld des Bildschirmformats zu der Attributwertkombination aus dst-format - auf jeden Fall aber mit gesetzter Schreibmarke.

(3b) bzgl. der im Datenstationsformat definierten Attributkategorien enthalten die Attributwertkombinationen der reinen E-Felder die entsprechenden Attributwertkombinationen aus dst-format.

Die Unterscheidung, ob ohne oder mit Attributwertangaben gearbeitet wird, ist fest mit einem entsprechenden Merkmal der aufgerufenen Formatgruppe verbunden, d.h. es gibt Formatgruppen ohne Attributwertangaben in den Ausgabenachrichten und Format-

gruppen mit Attributwertangaben in den Ausgabenachrichten. Die Formatkennzeichen der zugehörigen Formate lassen das Merkmal erkennen: im ersten Fall ist das erste Zeichen des Formatkennzeichens ein *, im zweiten Fall ein +.

c.3) Änderungen des Bildschirmformats durch den Benutzer

Nach Beendigung der Ausgabe kann der Benutzer die Zeichenreihen in nicht geschützten, nicht auswählbaren Feldern ändern, und er kann auswählbare Felder auswählen. Durch eine Schreiboperation in einem Feld ändert sich ggf. auch der Attributwert zum Änderungszustand in 'verändert', durch Auswählen eines Feldes ändert sich ggf. der Attributwert zum Auswahlzustand in 'ausgewählt' (das 'ggf.' weist lediglich darauf hin, daß ein verändertes bzw. ausgewähltes Feld bereits den Attributwert 'verändert' bzw. 'ausgewählt' haben kann).

Schließlich kann der Benutzer signalisieren, daß aus dem Bildschirmformat nunmehr eine Eingabenachricht gebildet und an den Dialogvorgang übermittelt werden soll. Damit verliert er seine Eingabeberechtigung bis zu dem Zeitpunkt, an dem er die nächste Ausgabenachricht vollständig erhalten hat.

c.4) Erzeugen einer Eingabenachricht aus einem Bildschirmformat

Eine Eingabenachricht e-nachricht zu einem Eingabenachrichtensformat e-format mit den Felddatenangaben feldlänge₁, ..., feldlänge_r besteht aus einer Folge von r Zeichenreihen der entsprechenden Längen. Die Zeichenreihen aus e-nachricht, die Literalfeldern von e-format entsprechen, sind die entsprechenden Literale aus e-format.

Aus einem Bildschirmformat e-bildschirmformat entsteht folgende Eingabenachricht gemäß dst-format und e-format.

- (1) Für die Literalfelder entsprechenden Felder gilt das oben Gesagte.
- (2) Für die Zeichenreihen in Feldern, die auswählbaren Feldern von e-bildschirmformat entsprechen, gilt: falls ein solches Bildschirmfeld den Attributwert 'ausgewählt' (bzw. 'nicht ausgewählt') hat, wird eine Zeichenreihe der benötigten

Länge aus lauter INIT2-Zeichen (bzw. INIT3-Zeichen) in das entsprechende Feld von e-nachricht eingesetzt.

- (3) Für die Zeichenreihen in Feldern, die nicht-auswählbaren Feldern von e-bildschirminhalt entsprechen, gilt: falls das Bildschirmfeld den Attributwert 'verändert' hat, so wird in das entsprechende Feld von e-nachricht die Zeichenreihe aus dem Bildschirmfeld eingesetzt; andernfalls wird eine Zeichenreihe der benötigten Länge aus lauter INIT1-Zeichen eingesetzt.

Eine Überführung von Attributwertkombinationen vom Bildschirminhalt in die Eingabenachricht findet zwar nicht statt, falls aber die Ausgabenachricht mit Attributwertkombinationen ausgegeben wird, so muß auch die folgende Eingabenachricht mit Angabe eines +-Zeichens in der ersten Zeichenposition der Formatangabe gelesen werden. Die Folge ist, daß jedem Feldinhalt der Eingabenachricht eine undefinierte Attributwertkombination (im Arbeitsspeicher: 2 Byte undefinierten Inhalts) vorangestellt wird.

d) Darstellung formatierter Nachrichten im Arbeitsspeicher

Zum Transport an der TAS-Kernschnittstelle werden formatierte Nachrichten als Bytevektoren im Arbeitsspeicher dargestellt. Sie werden stets über den Datenbereich eines KA übermittelt.

Sei Bytevektor(zr) die systemspezifische Ein-Byte-pro-Zeichen-Verschlüsselung einer Zeichenreihe zr und Zweibyteverschlüsselung(AWK) die systemspezifische Verschlüsselung einer Attributwertkombination AWK im Umfang von genau zwei Byte. Die leere Attributwertkombination (ohne Schreibmarke) hat immer das 16-stellige Binärwort aus lauter Nullen als Zweibyteverschlüsselung.

Sind $zr_1, zr_2, \dots, zr_p$ die in Länge und Anordnung den Feldinhalten entsprechenden Zeichenreihen einer formatierten Nachricht ohne Attributwertkombinationen, so ist der sie repräsentierende Bytevektor einfach Bytevektor(zr_1) unmittelbar gefolgt von Bytevektor(zr_2) usw. unmittelbar gefolgt von Bytevektor(zr_p).

Beim Plusformat wird für eine Nachricht $AWK_i, zr_i, AWK_2, zr_2, \dots, AWK_p, zr_p$ (AWK_i sei die jeweilige Attributwertkombination) durch den Bytevektor, der aus den Bestandteilen

Zweibyteverschlüsselung(AWK_i) unmittelbar gefolgt von
Bytevektor(zr_i) $i = 1, 2, \dots, p$

durch nahtlose Aneinanderreihung entsteht.

Anmerkung: Bei den Eingabenachrichten sind die Inhalte der Zweibyteverschlüsselung(AWK_i) undefiniert.

Die Zweibyteverschlüsselung der Attributwertkombinationen ist in den Anwendungsprogrammen durch eine Tabelle verfügbar, auf die mit Hilfe bekannter (nach gewissen Regeln aus den Attributwerte darstellenden Zeichenreihen gebildeten) Buchstabenfolgen zur Bezeichnung bestimmter Attributwertkombinationen (vgl. Tabelle 8) zugegriffen wird. Das Setzen der Schreibmarke in einer gegebenen Zweibyteverschlüsselung wird durch Aufruf eines eingebauten Unterprogramms KDCSCUR erreicht, das mit der linken Byteadresse der zu modifizierenden Zweibyteverschlüsselung zu versorgen ist.

e) Mehrteilige Nachrichten

Besteht eine formatierte Ausgabenachricht aus mehreren Nachrichtenanteilen, so unterliegt jeder Nachrichtenteil demselben Ausgabenachrichtenformat und damit derselben Formatgruppe. Jeder Nachrichtenteil wird gemäß den oben angegebenen Regeln für einteilige Nachrichten durch Ausgabe in einen vollständigen Bildschirminhalt umgesetzt. Der Benutzer bearbeitet einen solchen Bildschirminhalt und erzeugt dadurch einen Nachrichtenteil der folgenden Eingabenachricht.

Die folgende Eingabenachricht hat demnach genau so viele Nachrichtenteile wie die vorangehende Ausgabenachricht. Die inhaltlichen Zusammenhänge – soweit sie von der beteiligten Formatgruppe festgelegt bzw. ermöglicht werden, bestehen jeweils genau zwischen dem i-ten Nachrichtenteil der Ausgabenachricht und dem i-ten Nachrichtenteil der Eingabenachricht.

Sofern die Dialogdatenstation dies zuläßt, kann der Benutzer die einzelnen Nachrichtenteile mehrfach zur Bearbeitung aufrufen (Blättern), bevor er die gesamte Eingabenachricht zur Eingabe freigibt.

f) Details zu den zugehörigen TAS-Kern-Aufrufen

Eine Ausgabenachricht an die Dialogdatenstation des Vorgangs wird durch einen oder mehrere MPUT-Aufrufe (Funktionsausprägung NT oder NE) an den TAS-Kern übergeben. Die KA für den Nachrichtenaustausch mit der Dialogdatenstation stimmen in vielen Details mit den KA (gleichen Funktionsmerkmalen) für die Nachrichtentübergabe von PL an PL überein (vgl. Abschnitt 2.3.3.1.2). Nur sind hier die Nachrichten(-teile) formatiert und unterliegen den inhaltlichen Veränderungen an der Dialogdatenstation, bevor sie vom nachfolgenden PL wieder mittels MGET-Aufruf(en) übernommen werden. So kommt jetzt generell das Formatkennzeichen als weiterer Parameter hinzu (Kurzzeichen: kcnk); beim ersten MPUT-Aufruf gibt es darüberhinaus noch die Ausgabe-funktion als Parameter (Kurzzeichen: kcdf).

Eine beabsichtigte Ausgabe an die Dialogdatenstation wird dem TAS-Kern dadurch angezeigt, daß der Rufname der MPUT-Folge (vgl. Abschnitt 2.3.3.1.2) eine Folge von 8 Leerzeichen ist. Die für die Nachrichtenaussage wesentlichen MPUT-Aufrufe des PL bilden das relevante Anfangsstück aller seiner MPUT-Aufrufe gemäß Definition in Abschnitt 2.3.3.1.2. Der PL muß dann mit einem PEND-RE-Aufruf oder einem PEND-FI-Aufruf beendet werden. Der kcrp-Wert eines PEND-RE-Aufrufs ist notwendigerweise vom Wert im Rufnamenfeld der MPUT-Folge verschieden; denn er stellt einen gültigen Programmnamen für das als nächste zu startende DP dar. Der kcrn-Wert eines PEND-FI-Aufrufs ist bedeutungslos. Bei unzulässigen Kombinationen des Rufnamens der MPUT-Folge und der Funktionsausprägung des folgenden PEND-Aufrufs wird ein schwerwiegender Fehler mit kcsa-Wert = 82Z angezeigt.

Jeder von mehreren relevanten MPUT-Aufrufen übermittelt dem TAS-Kern genau einen Nachrichtenteil im Sinne der oben eingeführten mehrteiligen Nachrichten.

Erfolgte in dem PL überhaupt kein MPUT-Aufruf, so führt dies bei der Bearbeitung des PEND-RE- bzw. PEND-FI-Aufrufs zu einem schwerwiegenden Fehler und zum fehlerbehafteten Abbruch des Vorgangs (vgl. Abschnitt 2.3.6.1).

Das Formatkennzeichen, dessen erstes Zeichen für den hier behandelten Austausch formatierter Nachrichten stets + oder * sein muß, soll in der Folge der MPUT-Aufrufe stets dasselbe sein. Wechselt dieser Wert fehlerhafterweise, so wird dennoch das Format gemäß kcnk-Wert des ersten MPUT-Aufrufs mput₁ be-

nutzt. Ein MPUT-Aufruf mput_i, i > 1, mit kcnk(mput_i) ≠ kcnk(mput₁), erhält als Statusanzeige die Bemerkung kcsa-Wert = 05Z.

Der TAS-Kern ist nicht verpflichtet, das Formatkennzeichen eines MPUT-Aufrufs gegen den Bestand der eingeführten Formatkennzeichen zu prüfen. Der schwerwiegende Fehler 75Z (ungültiges Formatkennzeichen) tritt auf jeden Fall dann auf, wenn das Formatkennzeichen syntaktisch falsch (vgl. Definition in Abschnitt 2.2) ist.

Beim MPUT-Aufruf gibt es den für diese Art von TAS-Kernfunktion ('echte' Ausgaben) charakteristischen schweren Fehler "ungültiger Inhalt des Datenbereichs" (kcsa-Wert = 78Z). Er bedeutet, daß der Bytevektor im Datenbereich nicht im Sinne des genannten Nachrichtenformats und der allgemeineren Festlegungen über die Darstellungen der Attributwertkombinationen und die Darstellung der auszugebenden Zeichen interpretierbar ist. Eine mathematisch präzise Festlegung der Bedingungen, unter denen dieser Fehler auftritt, gibt diese Norm nicht.

Genau beim ersten MPUT-Aufruf in einem PL wird auch der Parameter Ausgabe-funktion (kcdf) ausgewertet. Es gibt z.zt. vier Ausgabe-funktionen:

- KCREPL (replace) : Löschen des Darstellungsbereichs
- KCERAS (erase) : Löschen der ungeschützten Felder. Dies ist nur sinnvoll, wenn das Datenstationsformat gegenüber dem des vorangehenden Aus-Eingabezyklus unverändert bleibt.
- KCALAR (alarm) : Auslösen eines akustischen Signals
- KCREPR (reproduce): Kopieren des Darstellungsbereichs auf einen der Datenstation zugeordneten Drucker. Bei dieser Ausgabe-funktion ist zunächst die mitangelieferte Ausgabenachricht gemäß Datenstationsformat im Darstellungsbereich darzustellen. Danach ist die geforderte Kopie anzufertigen.

Die Funktionsnamen aus sechs Großbuchstaben sind genormte Elemente der PL-Schnittstelle. Für die Anwendungsprogramme sind Anweisungen verfügbar, um die im Programmtext geschriebenen

Funktionsnamen in Verschlüsselungswerte umzusetzen, die dann als kcdf-Werte benutzbar sind. Falls dieser Parameter nicht mit einem der zulässigen Werte besetzt ist, wird ein Vorbesetzungswert - normalerweise KCREPL - benutzt; es erfolgt keine Fehlermeldung.

Die Übernahme der Eingabenachricht von der Dialogdatenstation im ersten DPL eines Dialogschritts erfolgt durch einen oder mehrere MGET-Aufrufe, in denen als Wert des Parameters kcnk das Formatkennzeichen des Eingabenachrichtenformats auftreten muß, das - gemäß der oben beschriebenen Formatkopplung - zu dem im vorangegangenen DPL benutzten Ausgabenachrichtenformat gehört. War dies ein +Formatkennzeichen (*Formatkennzeichen), so muß auch hier ein +Formatkennzeichen (*Formatkennzeichen) benutzt werden.

Verstöße gegen diese Vorschriften, die die Zusammenhänge zwischen einem Ausgabenachrichtenformat und dem nächsten Eingabenachrichtenformat betreffen, werden vom TAS-Kern nicht erkannt, d.h. es gibt keine diesbezüglichen Warnungen oder Fehlermeldungen. Im Sinne dieser Norm sind Auswirkungen solcher Verstöße undefiniert.

Jeder MGET-Aufruf übernimmt genau einen Nachrichtenteil in seinen Datenbereich der Länge kcdl. Die Darstellung des Nachrichtenteils folgt den oben angegebenen Festlegungen über die Darstellung formatierter Nachrichten im Arbeitsspeicher. Alle übrigen Details, insbesondere die Bedeutung und Überprüfung der Längenangaben und die Behandlung von Längenkonflikten, sind genauso geregelt wie für MGET-Aufrufe, die Nachrichtenteile vom Vorgänger-PL aufnehmen (vgl. Abschnitt 2.3.3.1.2). Ein Wechsel des Formatkennzeichens in einer Folge von MGET-Aufrufen wird entsprechend gehandelt wie in einer Folge von MPUT-Aufrufen. Auch hier ist im Fall des fehlerhaften Wechsels immer das Format gemäß Kennzeichen im ersten MGET-Aufruf wirksam.

2.3.5.2.2 Formatfreie Aufrufe und Aufrufe der FORMAT-Routine (MGET, MPUT)

Formatfreie MPUT- und MGET-Aufrufe und MPUT- und MGET-Aufrufe mit Aufruf der FORMAT-Routine mögen hier (MPUT-, MGET) Aufrufe ohne K-Formatdienste heißen. Für sie gilt, daß der kcnk-Wert des Aufrufs kein eigentliches Formatkennzeichen darstellt, d.h. nicht die Benennung eines vordefinierten Nachrichtenformats ist.

Der kcnk-Wert bei formatfreiem Aufruf hat als erstes Zeichen ein Leerzeichen; die übrigen sieben Bytewerte sind bedeutungslos. Das in Abschnitt 2.3.5.2.1 f) geforderte Gleichbleiben des Formatkennzeichens von MPUT-Aufrufen oder von MGET-Aufrufen reduziert sich hier auf die Forderung nach durchgängigem Auftreten des Leerzeichens in der 1. Byteposition des kcnk-Werts. Bei Verstößen gegen diese Regel wird im übrigen in der in Abschnitt 2.3.5.2.1 f) beschriebenen Weise verfahren.

Der kcnk-Wert beim expliziten Aufruf der FORMAT-Routine hat als erstes Zeichen ein Minuszeichen; die übrigen sieben Bytewerte sind aus der Sicht der PL-Schnittstelle beliebig und werden vom TAS-Kern nicht benutzt. Sie können aber für die FORMAT-Routine bedeutungsvoll sein. Das in Abschnitt 2.3.5.2.1 f) geforderte Gleichbleiben der kcnk-Werte in einer Folge von MPUT-Aufrufen oder von MGET-Aufrufen bei der Aus- oder Eingabe formatierter Nachrichten wird hier in jedem Detail genauso gefordert. Bei Verstößen wird entsprechend verfahren.

Die Formatroutine kann ihrerseits ein Formatkennzeichen (* oder + oder Leerzeichen in der erste Byteposition) erzeugen und dem TAS-Kern die Aufforderung hinterlassen, noch eine normale MPUT- oder MGET-Ausführung anzuschließen, bei der natürlich die Formatroutine nicht erneut aufgerufen werden darf.

Die Formatroutine kommt beim MPUT-Aufruf nicht zum Ablauf, falls einer der möglichen schwerwiegenden Fehler auftritt, abgesehen von dem Fehler "ungültiger Inhalt des Datenbereichs", der dann noch beim Ablauf der Formatroutine erkannt werden kann.

Nachrichtenaustausch ohne K-Formatdienste vollzieht sich im übrigen mit denselben KA-Folgen, wie sie für formatgebundene KA-Folgen beschrieben wurden. Die Nachrichtenteile sind hier unformatierte Zeichenreihen, d.h. Zeichenreihen, zu denen dem TAS-Kern keine Untergliederung bekannt ist.

MPUT-Aufrufe ohne K-Formatdienste verlangen gleichartige (formatfrei, FORMAT-Routine) MGET-Aufrufe im nachfolgenden PL. Bei Verstößen gegen diese Vorschrift gibt es keine Warnungen oder Fehlermeldungen, d.h. der TAS-Kern verhält sich hierbei analog wie beim Austausch formatierter Nachrichten.

Der schwerwiegende Fehler "ungültiger Inhalt des Datenbereichs" (vgl. Abschnitt 2.3.5.2.1 f)) kann auch bei formatfreier Ausgabe bzw. beim Aufruf der Formatroutine anläßlich eines MPUT-Aufrufs auftreten. Bei einem MGET-Aufruf kann dieser schwerwiegende Fehler höchstens dann auftreten, wenn es zu einem Aufruf der FORMAT-Routine kommt, d.h. sein Auftreten wird von der FORMAT-Routine festgestellt.

2.3.5.3 Freilaufende Ausgaben an Datenstationen (FPUT, PEND-RE, PEND-FI)

Ein DPL kann sog. freilaufende Ausgaben unter beliebigen Datenstationsruffamen zur Ausgabe an den TAS-Kern übergeben, ausgenommen als Zielangabe ist der Datenstationsruffame, der im KB-Kopf des DPL als Rufname seiner Dialogdatenstation eingetragen ist (Fehler mit kcsa-Wert = 412).

Anmerkung: Es ist nicht ausgeschlossen, daß unter einem zulässigen Datenstationsruffamen eine freilaufende Ausgabe des Dialogvorgangs später diejenige Datenstation erreicht, die seine Dialogdatenstation ist. Die Ausgabe kann allerdings sicher solange nicht erfolgen, wie diese Datenstation Dialogdatenstation des betrachteten Dialogvorgangs ist.

Der Vollzug der Ausgabe einer freilaufenden Nachricht ist niemals Voraussetzung für die Fortführung des Dialogs, aus dem sie hervorgeht. Es wird auch keinerlei Rückmeldung oder Quittung über den Vollzug an den aussendenden Vorgang produziert oder von diesem erwartet.

Freilaufende Ausgabenachrichten können frühestens ab Ende des Dialogschritts, aus dem heraus sie abgegeben wurden, ausgegeben werden. Mindestens bis dahin werden sie also im TAS-Kern zwischengepuffert (vgl. Abschnitt 2.3.6).

Anmerkung: Man beachte, daß der Terminus 'Nachricht' nicht grundsätzlich, sondern höchstens für den durchschnittlichen Anwendungsfall eine Größenordnung andeuten soll. Eine solche Nachricht kann hunderte von Zeilen umfassen, ggf. in mehreren Nachrichtenteilen.

Da verschiedene Vorgänge - zeitlich nicht aufeinander abgestimmt - freilaufende Ausgabenachrichten erzeugen können, werden sich gelegentlich vor Datenstationen Warteschlangen bilden. Hinzu kommt, daß eine Datenstation zeitweilig Dialogdatenstation eines Dialogvorgangs sein kann; in diesem Zustand können auf ihr überhaupt keine freilaufenden Nachrichten ausgegeben werden. Sie kann sogar aus noch schwächeren Gründen für freilaufende Ausgaben vorgänge gesperrt sein, z.B. wenn ein Datenstationsformat über das Ende eines Vorgangs hinaus an einer Datenstation erhalten bleiben soll (vgl. Abschnitt 2.3.1.1) oder wenn an der Datenstation ein Prädialog geführt wird.

Alle Nachrichtenteile einer mehrteiligen freilaufenden Ausgabenachricht müssen ohne Zwischenschieben irgendwelcher anderen freilaufenden Ausgabenachrichten an der fraglichen Datenstation ausgegeben werden.

Eine freilaufende Ausgabenachricht wird durch eine vollständige FPUT-Folge (vgl. Abschnitt 2.3.3.2) eines DPL, deren Zielangabe (Wert des Parameters kern) ein Datenstationsruffame ist, an den TAS-Kern übergeben. Es sind alle drei Aufrufarten (formatgebunden, formatfrei, FORMAT-Routine; vgl. Abschnitt 2.3.5.1) möglich. Bei formatierter Ausgabe ergeben sich für das Datenstationsformat, das an das aufgerufene Ausgabe-Nachrichtenformat gekoppelt ist, gegenüber den Bestandteilen nach Abschnitt 2.3.5.2.1 b) erhebliche Einschränkungen. Es fehlen in diesem Fall die Eingabefelder. Von den Attributkategorien sind die Angaben zur Veränderbarkeit, zur Auswählbarkeit und zum Eingabezwang nicht sinnvoll. Ebenso entfallen die Attributkategorien Veränderungsstatus und Auswahlstatus bzgl. der Felder des Darstellungsereichs. Bei Druckausgabe können auch einige der Attributwerte zur Darstellungsweise entfallen bzw. eine andere Interpretation erfordern.

Beim ersten FPUT-Aufruf einer vollständigen FPUT-Folge ist auch der Parameter kcdf zur Ansteuerung von Ausgabefunktionen nutzbar, vorausgesetzt, an der Datenstation lassen sich die Ausgabefunktionen in sinnvoller Weise interpretieren. Es gelten alle Details wie für den MPUT-Aufruf in Abschnitt 2.3.5.2.1 f) angegeben.

Der Fehler "ungültiger Inhalt des Datenbereichs" (vgl. Abschnitt 2.3.5.2.1 f) kann auch bei allen Varianten der FPUT-Aufrufe zur Ausgabe auf Datenstationen auftreten (kcsa-Wert = 482). Hier

ist er allerdings nur ein Fehler und kein schwerwiegender Fehler, weil angenommen wird, daß in einem Dialogvorgang die Situation mit dem Benutzer an der Dialogdatenstation erörtert und eventuell ein Weg gefunden werden kann, den Dialogvorgang sinnvoll weiterzuführen.

2.3.5.4 Sonderfunktionen der Eingabe (MGET)

Durch ein K-System wird die Verwendung von Funktionstasten an Eingabedatenstationen unterstützt. Das Betätigen einer Funktionstaste soll wie die Übergabe einer kurzen Eingabenachricht durch den Benutzer wirken. Im verarbeitenden Programmlauf soll erkennbar werden, daß eine bestimmte Funktionstaste betätigt wurde.

An der PL-Schnittstelle stehen dafür zwanzig Werte der Statusanzeige (kcsa-Werte 20Z, ..., 39Z) zur Verfügung, die bei einem ersten MGET-Aufruf (oder einem ersten FGET-Aufruf in einem FPL, vgl. Abschnitt 2.4.5) in einem PL zusammen mit der Längenangabe Null (kcla-Wert = 0) eindeutig anzeigen, daß eine Funktionstaste betätigt wurde (alle weiteren MGET-Aufrufe werden mit Statusanzeige 10Z beantwortet). Eine gleichzeitige Übermittlung einer normalen Eingabenachricht ist nicht möglich.

Für ein TAS wird eine gewisse Teilmenge T der Werte 20, 21, ..., 39 festgelegt, deren Elemente als mögliche Ergebnisse von Funktionstastenbetätigungen auftreten können sollen. Jede Funktionstaste der entsprechend ausgerüsteten eingabefähigen Datenstationen erhält einen Wert aus einem Intervall $I = [01, 02, \dots, nm]$, $nm < 20$. Die bei der Systemdefinition festgelegte Abbildung SC bildet das Intervall I (nicht notwendigerweise eindeutig) auf die vorgenannte Menge T ab. Wird eine Funktionstaste mit Nr. $i \in I$ betätigt, so ergibt dies, wenn $SC(i) = uv \in T$ ist, den kcsa-Wert = uvZ.

Falls gewisse Eingabegeräte nicht nm Funktionstasten haben oder nicht alle für Sonderfunktionen der Eingabe benutzt werden sollen, so kann folgende Ersatzdarstellung benutzt werden: man betrachte das erste (im Sinne des verwendeten Eingabedatenformats) echte Eingabefeld, in das der Benutzer in der gegebenen Situation etwas eintragen kann, das also insbesondere ungeschützt sein muß. Enthält dieses Feld in seinen ersten acht Zeichenpositionen (es muß mindestens acht Zeichenpositionen

haben) eine Zeichenreihe des syntaktischen Aufbaus

(KDCSxy) (inklusive Klammern!)

mit $xy \in \{01, 02, \dots, 20\}$, dann wirkt dies beim Absenden des Bildschirminhalts durch den Benutzer so, als sei eine Funktionstaste betätigt worden, die den Statuswert SC(xy)Z abliefern. Bei formatfreier Eingabe werden die ersten acht Zeichen der Eingabenachricht geprüft. Falls sie die Gestalt (KDCSxy) mit $xy \in \{01, 02, \dots, 20\}$ haben, wirkt dies ebenso.

Ist die Zeichenreihe zur Ersatzdarstellung syntaktisch inkorrekt, so wird der Bildschirminhalt situationsgemäß und regelgerecht in eine Eingabenachricht umgesetzt und übermittelt. Liegt xy in (KDCSxy) außerhalb von 01, ..., nm so wird als Warnung die Statusanzeige 19Z (nicht definierte Sonderfunktion) geliefert; darüberhinaus wird keine Eingabenachricht von der Dialogdatenstation geliefert (kcla-Wert = 0).

2.3.5.5 Das Anwendungssystem-Protokoll (LPUT)

Das Anwendungssystem-Protokoll - im folgenden kürzer Systemprotokoll genannt - dient den Vorgängen zur zentralen Protokollierung irgendwelcher Ereignisse, die in den Vorgängen registriert werden. Die einzelnen Einträge (Sätze), von denen jeder durch genau einen geeigneten KA in irgendeinem PL erzeugt wird, bilden im Systemprotokoll eine Folge, in der die Anordnung der Sätze näherungsweise der Anordnung der zugehörigen KA in der Zeit entspricht (genauere Aussagen weiter unten). Das Systemprotokoll ist für das TAS, in dem es erzeugt wird, ein nur beschreibbarer Speicher, dessen Inhalt dazu bestimmt ist, das System zu verlassen. Die Sätze im Systemprotokoll können von den Vorgängen in keiner Weise wieder gelesen werden.

Anmerkung: Es ist andererseits durchaus möglich, daß Teile des Systemprotokolls - etwa Ausschnitte über gewisse Zeitintervalle - durch Eingabeaktionen wieder in das TAS zurückkehren.

Aus den genannten Gründen gehört das Systemprotokoll begrifflich zu den Ausgabeeinrichtungen, wenn es auch nicht mit einer bestimmten Ausgabedatenstation in Verbindung gebracht werden kann.

Auf das Systemprotokoll wird von den verschiedenen Vorgängen asynchron zugegriffen. Die Sequentialisierung der Zugriffe - Anhängen von Sätzen oder ganzen Satzfolgen - leistet der TAS-Kern. Alle Beiträge zum Systemprotokoll, die innerhalb eines Dialogschritts an den TAS-Kern übergeben werden, werden bis zum Ende dieses Dialogschritts gesammelt und im Zuge der dann notwendigen Abschlußarbeiten (vgl. auch Abschnitt 2.3.6) als zusammenhängende Satzfolge in das Systemprotokoll übertragen. In dieser Satzfolge entspricht die Anordnung genau der zeitlichen Reihenfolge der Übergeben der Sätze an den TAS-Kern durch KA.

Ein PL übergibt dem TAS-Kern durch einen KA mit Funktionsmerkmal LPUT (aus engl. put log) seinen Beitrag zu genau einem Satz des Systemprotokolls im Datenbereich des Aufrufs als Bytevektor. Die Längenangabe findet sich - wie üblich - unter dem Parameter kcdl. Die Länge des Datenbereichs ist durch den Systemparameter MAX_LÄNGE_SYSPROT_SATZ (vgl. Abschnitt 2.2) beschränkt. Die Länge LDAT des Bytevektors, der aus dem Datenbereich in den Satz im Systemprotokoll übernommen wird, ist also

$$LDAT = \min(kcdl\text{-Wert}, MAX_LÄNGE_SYSPROT_SATZ)$$

wobei die ersten LDAT Byte-Werte aus dem Datenbereich übernommen werden. Dieser übernommene Bytevektor werde als der explizite PL-Beitrag im Satz des Systemprotokolls bezeichnet. Falls $kcdl\text{-Wert} > MAX_LÄNGE_SYSPROT_SATZ$ ist, erhält der PL die Bemerkung 01Z als Statusanzeige.

Der Satz, der aufgrund eines KA in das Systemprotokoll kommt, enthält zusätzlich zum expliziten PL-Beitrag noch den Inhalt des KB-Kopfes des Vorgangs zum Zeitpunkt des Aufrufs. In einem Satz des Systemprotokolls steht der kopierte Inhalt des KB-Kopfes vor dem expliziten PL-Beitrag.

2.3.6 Der Dialogschritt als Transaktion und Sicherungsabschnitt (PEND-RE, PEND-FI, PEND-ER)

2.3.6.1 Die Konsequenzen der Alles-oder-Nichts-Regel für die Arbeitsweise des TAS-Kerns

Die Alles-oder-Nichts-Regel betrifft zunächst einmal die Inhalte der Anwendungsprogramme. Sie müssen in den Dialogschritten sinnvolle (im Sinne der inhaltlichen Zielsetzung des TAS) Konstellationen von Änderungen an den globalen Daten sowie von

Ausgabenachrichten hervorbringen, auf die dann die Alles-oder-Nichts-Regel anzuwenden ist.

Dies bedeutet u.a., daß innerhalb eines Dialogschrittes kein freilaufender Ausgabeauftrag gegeben werden darf, von dem sich später im selben Dialogschritt - trotz fehlerfreien Funktionierens der PL und des TAS-Kerns - noch herausstellen kann, daß er nicht hätte gegeben werden dürfen; denn es gibt dann keine Möglichkeit mehr an der PL-Schnittstelle, bereits dem TAS-Kern übergebene freilaufende Ausgabeaufträge selektiv zu annullieren. Ein PL kann dann höchstens mittels eines expliziten PENDER-Aufrufs den gesamten Dialog beenden und damit auch die anstehenden Ausgabeaufträge löschen. Dieses Vorgehen kann aber für den Benutzer einen mühsamen erneuten Anlauf bedeuten.

Mit den oben angedeuteten anwendungsspezifischen Aspekten der Alles-oder-Nichts-Regel beschäftigt sich der TAS-Kern direkt nicht. Andererseits muß er zur Realisierung der Alles-oder-Nichts-Regel beitragen, wenn es darum geht, die Wirkung von Fehlern abzufangen, die in den PL nicht erkannt oder in den PL nicht behandelt werden können.

Nach Abschnitt 2.1.1 ist in Dialogen jede Interaktion eine Transaktion. Wegen der Alles-oder-Nichts-Regel bedeutet dies, daß der TAS-Kern die von den PL durch KA definierten beabsichtigten globalen Wirkungen (vgl. Abschnitt 0.2) frühestens bei der Bearbeitung des PENDER-Aufrufs am Ende des Dialogschritts in dem Sinne vollziehen darf, daß sie außerhalb des erzeugenden Vorgangs erkennbar und wirksam werden - nämlich für andere Vorgänge und für die Benutzer. Es sei daran erinnert, daß dieser Schlußfolgerung über die Arbeitsweise des TAS-Kerns zwei weitere Annahmen zu Grunde liegen. Die erste lautet, daß es eine nie ganz auszuschließende Gefahr von Behinderungen und Störungen auf dem jeweils verbleibenden Weg eines Vorgangs bis zum nächsten Dialogschrittlende gibt. Die zweite Annahme ist, daß (globale) Wirkungen auf andere Vorgänge und insbesondere auf Instanzen außerhalb des Systems i.a. nicht wieder rückgängig gemacht werden können. Es gibt dann in der Tat noch die weitergehende Annahme, daß die beabsichtigten globalen Wirkungen aus einem Dialogschritt nach einem fehlerfreien PENDER-Aufruf am Dialogschrittlende vom TAS-Kern und eventuell weiteren beteiligten Funktionseinheiten (z.B. freilaufenden Vorgängen) vollständig und fehlerfrei vollzogen werden.

Zu der notwendigen Verzögerung der Außenwirksamkeit aller beabsichtigten globalen Wirkungen kommt die Forderung hinzu, daß diese Verzögerung aus der Sicht des erzeugenden Dialogvorgangs selbst (und damit insbesondere aus der Sicht der Programmierer der Anwendungsprogramme) – also im Innenverhältnis – überhaupt nicht oder nur in sehr speziellen Situationen, z.B. beim programmierten expliziten Fehlerabbruch eines Dialogschritts und damit des Vorgangs, sichtbar werden soll.

Die globalen Wirkungen eines Dialogschritts lassen sich wie folgt einteilen:

- Veränderungen von Inhalten globaler Speicher
- Startwünsche für freilaufende Ausgaben und Beiträge zum Systemprotokoll
- Startwünsche für (FP-gestützte) freilaufende Vorgänge
- Ausgabe einer Nachricht an die Dialogdatenstation des Vorgangs.

Für die letzten drei Einteilungspunkte läßt sich der Verzögerungseffekt einfach dadurch erzielen, daß diese geforderten globalen Wirkungen mindestens bis zum Dialogschritts im TAS-Kern zwischengespeichert werden. Dies wird gegenüber den PL verdeckelt gehalten; denn sie können auf diese zwischengespeicherten Objekte (Auftragsbeschreibungen) in keiner Weise zugreifen, es sei denn, daß sie sie kollektiv vernichten durch einen PEND-ER-Aufruf (Näheres weiter unten). Wird das Dialogschrittsende nicht fehlerfrei erreicht und durchschritten, so werden die für diesen Vorgang beim TAS-Kern zwischengespeicherten Aufträge einfach gelöscht.

Schwieriger liegt die Sache bei den globalen Speichern. Bei ihnen wird die Verzögerung der Außenwirkung von Zugriffen durch die ziemlich rigorose Sperrtechnik erreicht (vgl. Abschnitt 2.3.4.1) (außerhalb ist höchstens erkennbar, daß gewisse Satzsperrungen bestehen). Dies genügt aber für die Realisierung der Alles-oder-Nichts-Regel noch nicht. Falls der Dialogschritt nicht beendet werden kann, muß der Zustand der globalen Speicher wieder so hergestellt werden, als hätten die Veränderungen aus dem annullierten Dialogschritt nicht stattgefunden.

Konkreter bedeutet dies, daß die Benutzung der globalen Speicher innerhalb eines Dialogschritts keine Besonderheiten aufweisen darf, die auf irgendeine "Rücksetzfähigkeit" schließen ließen. Das bedeutet am Beispiel, daß ein innerhalb eines Dialogschritts einmal oder mehrmals beschriebener Satz beim späteren Lesen innerhalb des Dialogschritts stets den innerhalb des Dialogschritts zuletzt geschriebenen Inhalt aufweist oder daß ein im Dialogschritt gelöschter Satz beim späteren Lesezugriff im selben Dialogschritt als nicht-vorhanden gemeldet wird, während doch dieser Satz bei Annullierung dieses Dialogschritts sich in dem globalen Speicher wieder mit dem Inhalt vorfinden muß, den er unmittelbar vor dem ersten Zugriff aus diesem Dialogschritt heraus hatte.

Aussagen darüber, wie der TAS-Kern die Alles-oder-Nichts-Regel für die Zugriffe auf globale Speicher gewährleistet, sind nicht Bestandteil dieser Norm.

Bei einem fehlerfrei ausführbaren PEND-RE- oder PEND-FI-Aufruf werden die Veränderungen in den globalen Speichern und die Aufhebung der Satzsperrungen sofort ausgeführt, ebenso die Ausgabe der Nachricht an die Dialogdatenstation. 'Sofort' heißt hier nur, daß kein anderer Vorgang vorrangig auf die fraglichen Sätze oder die Dialogdatenstation zugreifen kann, so daß abschließend die Konkurrenz um anonyme Betriebsmittel die Ausführung zeitlich in nicht genau vorhersehbarer Weise beeinflussen kann. Anders ist es mit den freilaufenden Ausgaben und Vorgangstarts sowie mit den Eintragungen in das Systemprotokoll. Hier müssen die Aufträge ggf. in Warteschlangen eingereiht werden. Die Wirkungen, die im einzelnen bei endgültigem Vollzug der Aufträge erzielt werden, sind an anderer Stelle beschrieben.

Fehlerbehaftete Beendigung eines Dialogschritts

Jede fehlerbehaftete Beendigung eines Dialogschritts impliziert die gleichzeitige Beendigung des betroffenen Dialogvorgangs.

Auch wenn es zu einem PEND-RE- bzw. einem PEND-FI-Aufruf kommt, kann der Dialogschritt noch zu einer fehlerbehafteten Beendigung kommen, nämlich, wenn bei der PEND-Ausführung ein schwerwiegender Fehler erkannt wird. Außer den bei allen KA möglichen schwerwiegenden Fehlern (wie etwa ein fehlender INIT-Aufruf)

sind dies folgende:

- bei PEND-RE: der kcrn-Wert ist kein Programmrufrufname eines DP (kcsa-Wert = 74Z).
- bei PEND-RE und PEND-FI:
 - in den MPUT-Aufrufen des PL war als kcrn-Wert nicht die Zeichenreihe aus 8 Leerzeichen angegeben. (kcsa-Wert = 82Z)
 - in dem PL wurde überhaupt kein MPUT-Aufruf ausgeführt (kcsa-Wert = 83Z).

Bei den fehlerbehafteten Beendigungen eines Dialogschritts sind grundsätzlich zwei Kategorien von Anlässen zu unterscheiden:

- a) programmierte explizite PEND-ER-Aufrufe
- b) vom TAS-Kern erkannte Fehler, die zum Abbruch führen:
 - b 1) ein PL erhält sich schwerwiegend fehlerhaft
 - b 1.1) bei einem KA
 - b 1.2) außerhalb eines KA
 - .(z.B. Speicherschutzalarm, arithmetischer Alarm, Zeitüberschreitung)
 - b 2) außerhalb der Verantwortung der PL tritt ein Fehler auf, der zum Abbruch eines Vorgangs Anlaß gibt und bei dem auch ein Wiederanlauf (vgl. Abschnitt 2.3.6.2) keinen Erfolg verspricht.

Die Ausführung der PEND-ER-Aufrufe umfaßt folgende Aktionen: alle beabsichtigten globalen Wirkungen des Dialogschritts werden annulliert. Die EXIT-Routine, ein installationspezifisches, systemglobales Programm, wird zum Ablauf gebracht, wobei dieser Programmablauf (der nicht auf der PL-Schnittstelle abläuft) u.a. zum gesamten Arbeitsspeicher des zu beendenden PL Zugriff hat. Der Ablauf der EXIT-Routine wird in der Regel gewisse Auszüge aus dem Arbeitsspeicher des PL herstellen, die zwecks Fehleranalyse das System verlassen über Ausgabewege, die nicht zu den genormten TAS-Kernschnittstellen gehören. Danach hat der PEND-ER-Aufruf noch alle Wirkungen eines fehlerfreien PEND-FI-Aufrufs mit der Abweichung, daß statt einer im Dialogschritt durch MPUT-Aufrufe bereitgestellten Nachricht eine sy-

stemglobale, d.h. keine Information über die spezielle Fehler-situation enthaltende Nachricht an die Dialogdatenstation ausgegeben wird.

Bei allen Anlässen der Kategorie b) wird zunächst die EXIT-Routine zum Ablauf gebracht, die Zugang zum Arbeitsspeicher des zu beendenden PL und dadurch auch zu den Parameterwerten des KA hat, falls der Abbruch wegen des Auftretens eines schwerwiegenden Fehlers bei einem KA erfolgt. Danach ist der Ablauf wie bei einem expliziten PEND-ER-Aufruf, nur daß der dort vorgesehene Aufruf der EXIT-Routine nicht (noch einmal) stattfindet.

2.3.6.2 Der Dialogschritt als Sicherheitsabschnitt

Die geforderte hohe Stabilität eines TAS und hier speziell eines TAS mit einem TAS-Kern soll u.a. dadurch erzielt werden, daß auf gewisse, vom TAS-Kern erkannte Situationen, in denen ein Dialogvorgang beschädigt worden ist oder behindert (z.B. durch Verklemmungen) wird, mit einem Wiederanlauf des betroffenen Dialogvorgangs beim Beginn des laufenden Dialogschritts reagiert wird.

Ein Wiederanlauf eines Dialogvorgangs bei Beginn des laufenden Dialogschritts bedeutet, daß alle bereits beim TAS-Kern manifesten beabsichtigten globalen Wirkungen des Dialogschritts annulliert werden und daß der interne Zustand des Vorgangs, der bei Beginn des laufenden Dialogschritts bestand, - natürlich unter Aufgabe des aktuell erreichten internen Zustands - wiederhergestellt wird und dadurch der Dialogschritt erneut - i.a. in einem veränderten Systemzustand außerhalb des Vorgangs - zum Ablauf gebracht wird.

Anmerkung: Der interne Zustand eines Dialogvorgangs bei Beginn eines Dialogschritts besteht aus seiner lokalen Ausführungs-umgebung (KB, LSSB, Programmrufrufname des nächsten PL, ev. aufgerufene Formatgruppe) und der kompletten Eingabenachricht von der Dialogdatenstation, deren Eintreffen das Startereignis für diesen Dialogschritt war.

Im KB-Kopf bleiben Datum und Uhrzeit des Vorgangsstarts beim Wiederanlauf unverändert. Der Start des ersten Programmlaufs des Dialogschritts wird erneut vollzogen, wobei insbesondere seine Startzeit im KB-Kopf aktualisiert, d.h. die Uhrzeit des beim Wiederanlauf neu vollzogenen Starts eingetragen wird.

Ein Wiederanlauf soll genau dann erfolgen, wenn es sicher oder hinreichend wahrscheinlich ist, daß der so herbeigeführte erneute Ablauf des Dialogschritts nicht die gleiche Beschädigung oder Behinderung erfährt, die beim vorangehenden Ablauf den Wiederanlauf veranlaßt hat.

Die Norm macht keine Aussagen darüber, wie der TAS-Kern die oben beschriebene Aufgabe (Wiederanlauf von Dialogschritten) bewältigen soll.

2.3.6.3 Integriertes Datenbanksystem

Falls das betrachtete TAS ein integriertes Datenbanksystem (vgl. Anmerkung am Schluß von Abschnitt 0.3) umfaßt oder andere globale Speicher (neben den in der PL-Schnittstelle enthaltenen), so beziehen sich die Aussagen über die Maßnahmen zur Realisierung der Alles-oder-Nichts-Regel sowie die Forderung, daß der Dialogschritt Sicherheitsabschnitt sein soll, auch auf die Datenbankzugriffe oder die Zugriffe zu den anderen globalen Speichern. Diese Zugriffe sind der Liste der (beabsichtigten) globalen Wirkungen eines Dialogschritts hinzuzufügen.

2.3.7 Fehlerfreie Beendigung eines Dialogvorgangs

Ein KA der TAS-Kernfunktion PEND-FI beendet einen Dialogvorgang, wobei gleichzeitig der letzte Dialogschritt beendet wird. Außer den Wirkungen des Dialogschrittes (vgl. u.a. Abschnitte 2.3.4.1, 2.3.5.2 und 2.3.6.1) wird die gesamte lokale Ausführungsumgebung des Dialogvorgangs – insbesondere der LSSB und der KB – gelöscht. Die Dialogdatenstation geht in den Zustand 'Keinem Dialog zugeordnet' über.

Der bei den meisten anderen PEND-Aufrufen wichtige Parameter kern (Programmrufrufname der Folge DPL) ist beim PEND-FI-Aufruf bedeutungslos.

Zu Fehlern beim PEND-FI-Aufruf vgl. Abschnitt 2.3.6.1 (Fehlerbehaftete Beendigung eines Dialogschritts).

2.4 Freilaufende Vorgänge und die FPL-Schnittstelle

Vorbemerkung: Freilaufende Ausgaben (vgl. Abschnitte 2.4.5.3 und 2.4.5.2) führen zu freilaufenden Vorgängen, die vollständig im TAS-Kern abgewickelt werden. Diese freilaufenden Vorgänge werden im folgenden nicht weiter betrachtet, sondern ausschließlich die programmgestützten freilaufenden Vorgänge, d.h. solche, deren Inhalt ganz wesentlich durch den Ablauf eines bestimmten FP festgelegt wird.

2.4.1 Anforderung der Eröffnung und Grobstruktur eines freilaufenden Vorgangs

2.4.1.1 Anforderung und Eröffnung eines freilaufenden Vorgangs (FGET, FPUT)

Ein freilaufender Vorgang kann durch zwei grundsätzlich verschiedene Arten von Aktionen angefordert werden:

- eine geeignete Eingabe an einer eingabefähigen und aktuell keinem Dialogvorgang zugeordneten Datenstation.
- durch einen oder mehrere zusammengehörige geeignete FPUT-Aufrufe in einem DPL (vgl. Abschnitt 2.3.3.2) oder in einem FPL.

Die beiden Arten der Anforderung werden nacheinander behandelt.

Anforderung der Eröffnung durch Eingabe an einer Datenstation

Hierbei besteht weitreichende Übereinstimmung mit den Maßnahmen bei der Anforderung der Eröffnung eines Dialogvorgangs (vgl. Abschnitt 2.3.1.1). Die im Prädialog bereitzustellenden Datenobjekte sind auch hier ein Programmrufrufname, eine Eingabenachricht und eine Benutzeridentifikation, die auch fehlen kann. Der Programmrufrufname ist notwendigerweise der Programmrufrufname eines FP; denn diese Tatsache kennzeichnet die Anforderung der Eröffnung eines freilaufenden Vorgangs. Dieser Programmrufrufname ist der Name des ersten (und im übrigen einzigen; vgl. Abschnitt 2.4.1.2) in dem zu eröffnenden Vorgang zu startenden FP.

Die Aussagen von Abschnitt 2.3.1.1 über die Übernahme der Eingabenachricht gelten auch für den FPL mit dem einen Unter-

Für die Eröffnung eines freilaufenden Vorgangs auf Grund einer vollständigen FPUT-Folge in einem PL liegen dann folgende Datenobjekte vor:

- der Programmrufofname eines FP.
- eine (unformatierte) Eingabenachricht ggf. gegliedert in mehrere Nachrichtenteile.

Ausgehend von diesen Datenobjekten wird ein freilaufender Vorgang in genau derselben Weise eröffnet wie im Falle einer entsprechenden Eingabe von einer Datenstation - mit Abweichung in folgenden Details:

Regel (Vererbung)
Die Positionen "Datenstationsrufofname" und "Datenstationstyp" sowie die Position "Benutzeridentifikation" werden mit den Inhalten der entsprechenden Positionen des KB-Kopfs des anfordernden Vorgangs besetzt.

Diese Regelung bedeutet, daß die genannten drei Datenobjekte im KB-Kopf - von denjenigen Vorgängen ausgehend, deren Eröffnung direkt von einer Datenstation aus angefordert wird - entlang einer Kette von Eröffnungsanforderungen "vererbt" werden.

2.4.1.2 Grobstruktur eines freilaufenden Vorgangs (PEND-FI)

Ein freilaufender Vorgang - gleichgültig in welcher Weise seine Eröffnung angefordert wurde - hat keine ihm zugeordnete Datenstation. Falls die Vorgangseröffnung von einer Datenstation aus angefordert wird, so kehrt diese nach Beendigung der Eingabe der Anforderung sofort wieder in einen neutralen Zustand (an keinen Vorgang gebunden; frei für beliebige weitere Verwendung) zurück.

Der schließlich eröffnete freilaufende Vorgang besteht aus genau einem FPL. Falls dieser FPL durch einen PEND-Aufruf beendet wird, so kann dies nur ein PEND-FI- oder ein PEND-ER-Aufruf sein. Beide Aufrufe haben keinen ergänzenden Parameter.

Aussagen zum Transaktionscharakter eines FPL bzw. eines freilaufenden Vorgangs und zu seiner fehlerfreien Beendigung finden sich in den Abschnitten 2.4.6 und 2.4.7.

schied, daß die Eingabenachricht nicht mittels MGET-Aufrufen sondern mittels formal und inhaltlich gleichbedeutenden FGET-Aufrufen übernommen wird. Auch Funktionstasten können zur Anforderung der Eröffnung eines freilaufenden Vorgangs benutzt werden (vgl. Abschnitt 2.3.5.4). Auch die Fehlerbedingungen und die zugehörigen Statusanzeigen sind dieselben, nur daß hier MGET, dort FGET ein ungültiges Funktionsmerkmal ist.

Die in Abschnitt 2.3.1.1 besprochene Problematik bei der Benutzung eines Datenstationsformats im Prädialog und eines dazu passenden Eingabenachrichtenformats besteht hier in gleicher Weise.

Bei der Eröffnung eines freilaufenden Vorgangs durch den TAS-Kern wird der KB-Kopf wie bei der Eröffnung eines Dialogvorgangs besetzt - mit einer Ausnahme:

- die Position 'Programmlauf im Vorgang' (KB[32]; Kurzzeichen: kcpv) wird mit dem Buchstaben A besetzt (mnemotechnisch für 'asynchron', eine nicht mehr gebräuchliche Bezeichnung für 'freilaufend').

Der im KB-Kopf verzeichnete Rufname einer Datenstation und die zugehörige Datenstationstypangabe gehören hier zu denjenigen Datenstation, an der die Anforderung zu dieser Vorgangseröffnung eingegeben wurde.

Für einen freilaufenden Vorgang wird bei seiner Eröffnung ein Lokaler Sekundärer Speicherbereich (LSSB) eingerichtet.

Der gestartete FPL stellt eine vollständige Transaktion dar (vgl. insbesondere Abschnitt 2.4.6).

Anforderung der Eröffnung eines freilaufenden Vorgangs durch Aktionen eines Programmlaufs

Wie in Abschnitt 2.3.3.2 im Detail ausgeführt kann ein DPL mittels vollständiger FPUT-Folgen dem TAS-Kern Anforderungen der Eröffnung freilaufender Vorgänge übergeben. Dieselben Aufrufen folgen mit gleichartiger Wirkung sind auch in FPL möglich (vgl. auch Abschnitt 2.4.3).

2.4.2 Programmlauflokale Speicher

2.4.2.1 Der Arbeitsspeicher (INIT)

Hinsichtlich Aufbau und Nutzung des Arbeitsspeichers gelten alle Aussagen aus den entsprechenden Abschnitten (alle Abschnitte von Abschnitt 2.3.1) für DPL auch für FPL, abgesehen von den Unterschieden bei der Vorbesetzung einiger Positionen im KB-Kopf, die in Abschnitt 2.4.1.1 beschrieben sind.

Anmerkung: Der KB-Programmbereich muß - sofern er eine Länge größer Null hat - bei Vorgangseröffnung notwendigerweise einen undefinierten Inhalt haben.

2.4.2.2 Der Lokale Sekundäre Speicherbereich (SPUT, SGET, SREL)

Für freilaufende Vorgänge hat der LSSB, dessen Zugriffsdetails an der DPL- und der FPL-Schnittstelle völlig identisch geregelt sind, ausschließlich den Charakter eines programmlauflokalen Speichers, in dem Zwischenergebnisse gehalten werden, die nicht im Arbeitsspeicher gehalten werden können oder sollen.

2.4.3 Systeminterne Datenübergabe von Programmlauf zu Programmlauf (FPUT, FGET)

Vorbemerkung: Unter Datenübergabe von PL zu PL wird hier nur ein gezielter Datentransport verstanden, bei dem der Empfänger-PL auch für den TAS-Kern erkennbar ist. Die Datenübergabe auf dem Weg über globale Speicher bleibt außer Betracht.

Da ein FPL einen vollständigen Vorgang darstellt, ist gezielte Datenübergabe zwischen PL, an der mindestens ein FPL beteiligt ist, stets ein vorgangsübergreifender Datentransport. Die einzige Form einer solchen Datenübergabe ist die bereits in Abschnitt 2.3.3.2 beschriebene (von Formatkennzeichen und Formatfreien) Nachrichtenübergabe von einem DPL an einen von ihm angeforderten FPL bzw. von einem FPL an einen von ihm angeforderten FPL. Die FPUT-Aufrufen der FPL-Schnittstelle haben exakt die gleichen Wirkungen wie an der DPL-Schnittstelle. Der Verzögerung des Starts der angeforderten FPL bis zum Dialogschritende entspricht hier eine Verzögerung mindestens bis zum Ende des anfordernden FPL und damit des anfordernden freilaufenden Vorgangs (vgl. insbesondere auch Abschnitt 2.4.6).

2.4.4 Globale Speicher (SPUT, SGET, SREL, PTDA, GTDA)

Die globalen Speicher (der Globale Sekundäre Speicherbereich und die Datenstationsspeicherbereiche) sind für FPL ebenso zugänglich wie für DPL und zwar mit Hilfe derselben TAS-Kernaufforderungen (vgl. Abschnitt 2.3.4 mit Unterabschnitten) - mit einer einzigen Abweichung bzgl. der Zugänglichkeit und Adressierung der Datenstationsspeicherbereiche: ein FPL kann auf jeden beliebigen DSB zugreifen. Bei einem GTDA- bzw. PTDA-Aufruf in einem FPL muß daher - zusätzlich zu den Parametern, die auch in DPL benötigt werden (vgl. Abschnitt 2.3.4.3) - im Parameter kcdn der Datenstationsrufname angegeben werden, der den zugehörigen DSB kennzeichnet. Ein kcdn-Parameterwert, der kein eingeführter Datenstationsrufname ist, führt zu einem Fehler mit kcsa-Wert = 46%.

Eine Satzsperrung zu Gunsten eines zugreifenden freilaufenden Vorgangs wird beim zeitlich ersten Zugriff auf einen Satz gesetzt und bleibt dann genau bis zum Ende des freilaufenden Vorgangs bestehen (vgl. auch Abschnitt 2.4.6).

2.4.5 Nachrichtenaustausch mit Datenstationen / Systemprotokoll

2.4.5.1 Eingabenachrichten (FGET)

Eine Eingabenachricht von einer Datenstation an einen freilaufenden Vorgang bzw. zur Bearbeitung in einem FPL kann nur diejenige Eingabenachricht sein, die bei der Anforderung der Eröffnung eines freilaufenden Vorgangs von dieser Datenstation aus eingegeben wird (vgl. Abschnitt 2.4.1.1).

2.4.5.2 Ausgabenachrichten (FPUT)

Da einem freilaufenden Vorgang keine Datenstation zugeordnet ist, kann eine Ausgabe aus einem freilaufenden Vorgang nur zeitlich entkoppelt, d.h. in Gestalt einer freilaufenden Ausgabe erfolgen. Hierzu gilt in allen Details, was für die DPL-Schnittstelle in Abschnitt 2.3.5.3 beschrieben ist, mit folgenden Abweichungen:

- als Zielangabe (Parameter kcrn) kommen ausnahmslos alle eingeführten Datenstationsrufnamen in Betracht.