Festlegung der einheitlichen Schnittstelle für Zugriffsbeschränkung und andere digitale Fernsehrundfunkdecoder-Anwendungen

Deutsche Fassung EN 50221:1997 + Corrigendum:2000

DIN
EN 50221

ICS 33.160.25

Ersatz für
DIN EN 50221:1997-08

Common interface specification for conditional access and other digital video broadcasting decoder applications;
German version EN 50221:1997 + Corrigendum:2000

Spécification d'une interface commune pour l'accès conditionnel et d'autres applications dans un décodeur de télévision numérique;
Version allemande EN 50221:1997 + Corrigendum:2000

Die Europäische Norm EN 50221:1997, zusammen mit dem eingearbeiteten Corrigendum:2000, hat den Status einer Deutschen Norm.

Beginn der Gültigkeit

Die EN 50221 wurde am 1997-02-15 angenommen.

Das Corrigendum wurde im Februar 2000 herausgegeben.

Nationales Vorwort

Für die vorliegende Norm ist das nationale Arbeitsgremium K 742 „Audio-, Video- und Multimediasysteme, -geräte und -komponenten" der Deutschen Elektrotechnischen Kommission im DIN und VDE (DKE) zuständig.

Norm-Inhalt war veröffentlicht als E DIN EN 50221/A1:2000-06.

Änderungen

Gegenüber der DIN EN 50221:1997-08 wurden folgende Änderungen vorgenommen:

Die Abschnitte der eingearbeiteten Änderung korrigieren einen Fehler in der Information über das PC-Karten-Metaformat und passen den Text von DIN EN 50221:1997 so an, dass er mit der allgemeinen Industriepraxis der Verwendung von Interrupts für Modul-I/O übereinstimmt.

Frühere Ausgaben

DIN EN 50221: 1997-08

Das Corrigendum:2000 zu EN 50221:1997 wurde in den Text eingearbeitet und durch eine senkrechte Linie am linken Seitenrand im Text gekennzeichnet.

Fortsetzung Seite 2
und 79 Seiten EN

Deutsche Elektrotechnische Kommission im DIN und VDE (DKE)

Nationaler Anhang NA
(informativ)

Zusammenhang mit Europäischen und Internationalen Normen

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ist nachstehend wiedergegeben. Zum Zeitpunkt der Veröffentlichung dieser Norm waren die angegebenen Ausgaben gültig.

IEC hat 1997 die Benummerung der IEC-Publikationen geändert. Zu den bisher verwendeten Normnummern wird jeweils 60000 addiert. So ist zum Beispiel aus IEC 68 nun IEC 60068 geworden.

Tabelle NA.1

Europäische Norm	Internationale Norm	Deutsche Norm
EN ISO/IEC 13818-1:1997	ISO/IEC 13818-1:1996	DIN EN ISO/IEC 13818-1:1997-06
–	ISO/IEC 13818-9:1996	–
–	Normen der Reihe ISO/IEC 8824	–
–	ISO/IEC 8825:1990	DIN 66333:1993-09
–	ETS 300468:1995 zurückgezogen Ersatz EN 300468:1998	DIN ETS 300468:1996-05
–	EN 300468:1998	DIN EN 300468:1998-10
–	ETR 162:1995	–
–	PC Card Standard Volume 2	–
–	PC Card Standard Volume 3	–
–	PC Card Standard Volume 4	–
–	ETS 300743:1997	DIN ETS 300743:1998-06

Nationaler Anhang NB
(informativ)

Literaturhinweise

DIN 66333, *Informationstechnik – Kommunikation Offener Systeme; Basis-Codierungsregeln zur Notation Eins für darstellungsunabhängige Syntax; Identisch mit ISO/IEC 8825:1990.*

DIN EN ISO/IEC 13818-1, *Informationstechnik – Codierung von bewegten Bildern und damit verbundenen Toninformationen – Teil 1: Systeme (ISO/IEC 13818-1:1996); Englische Fassung EN ISO/IEC 13818-1:1997.*

DIN EN 300468, *Digitaler Fernsehrundfunk (DVB) – Festlegung der Serviceinformation (SI) für DVB-Systeme; Englische Fassung EN 300468 V 1.3.1 (1998-02).*

DIN ETS 300468, *Digitale Rundfunk-Systeme für Fernsehen, Ton und Datendienste – Festlegungen der Service-information (SI) für digitale Fernsehrundfunk-Systeme (DVB); Englische Fassung ETS 300468:1995.*

DIN ETS 300743, *Digitaler Fernsehrundfunk (DVB) – Untertitelungssysteme; Englische Fassung ETS 300743:1997.*

EUROPÄISCHE NORM
EUROPEAN STANDARD
NORME EUROPÉENNE

EN 50221

Februar 1997

ICS 33.160.25

Eingearbeitet Corrigendum Februar 2000

Deutsche Fassung

Festlegung der einheitlichen Schnittstelle für Zugriffsbeschränkung und andere digitale Fernsehrundfunkdecoder-Anwendungen

Common interface specification for conditional access and other digital video broadcasting decoder applications

Spécification d'une interface commune pour l'accès conditionnel et d'autres applications dans un décodeur de télévision numérique

Diese Europäische Norm wurde von CENELEC am 1997-02-15 angenommen und das Corrigendum im Februar 2000 herausgegeben.

Die CENELEC-Mitglieder sind gehalten, die CEN/CENELEC-Geschäftsordnung zu erfüllen, in der die Bedingungen festgelegt sind, unter denen dieser Europäischen Norm ohne jede Änderung der Status einer nationalen Norm zu geben ist.

Auf dem letzten Stand befindliche Listen dieser nationalen Normen mit ihren bibliographischen Angaben sind beim Zentralsekretariat oder bei jedem CENELEC-Mitglied auf Anfrage erhältlich.

Diese Europäische Norm besteht in drei offiziellen Fassungen (Deutsch, Englisch, Französisch). Eine Fassung in einer anderen Sprache, die von einem CENELEC-Mitglied in eigener Verantwortung durch Übersetzung in seine Landessprache gemacht und dem Zentralsekretariat mitgeteilt worden ist, hat den gleichen Status wie die offiziellen Fassungen.

CENELEC-Mitglieder sind die nationalen elektrotechnischen Komitees von Belgien, Dänemark, Deutschland, Finnland, Frankreich, Griechenland, Irland, Island, Italien, Luxemburg, Niederlande, Norwegen, Österreich, Portugal, Schweden, Schweiz, Spanien, der Tschechischen Republik und dem Vereinigten Königreich.

CENELEC

EUROPÄISCHES KOMITEE FÜR ELEKTROTECHNISCHE NORMUNG
European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique

Zentralsekretariat: rue de Stassart 35, B-1050 Brüssel

Vorwort

Diese Europäische Norm wurde von dem Technischen Komitee CENELEC TC 206 „Rundfunk-Empfangseinrichtungen“ ausgearbeitet.

Der Text des Entwurfs wurde dem Einstufigen Annahmeverfahren unterworfen und von CENELEC am 1997-02-15 als EN 50221 angenommen.

Nachstehende Daten wurden festgelegt:

– spätestes Datum, zu dem die EN auf nationaler Ebene durch Veröffentlichung einer identischen nationalen Norm oder durch Anerkennung übernommen werden muss (dop): 1997-10-01

– spätestes Datum, zu dem nationale Normen, die der EN entgegenstehen, zurückgezogen werden müssen (dow): 1997-10-01

Vorwort zu Corrigendum:2000

ANMERKUNG Die Ersatzabschnitte in diesem Corrigendum korrigieren einen Fehler in der Information über das PC-Karten-Metaformat und passen den Text von EN 50221:1997 so an, dass er mit der allgemeinen Industriepraxis der Verwendung von Interrupts für Modul-I/O übereinstimmt.

Inhalt

1 Einführung und Anwendungsbereich

Für den Digitalen Fernsehrundfunk wurde eine Reihe von Normen geschaffen. Diese Normen enthalten Quellencodierung, Kanalcodierung, Informationen über Dienste und Decoderschnittstellen. Wenn die Notwendigkeit besteht, den Zugriff auf einen Rundfunkdienst zu steuern, wird zusätzlich ein System zur Zugriffsbeschränkung (CA) benutzt. Es wurde entschieden, dass es nicht erforderlich ist, dieses System zu normen, obwohl ein einheitlicher Scrambling-Algorithmus zur Verfügung steht. Den Rundfunkbetreibern bleibt dann die Möglichkeit, Decoder mit unterschiedlichen Systemen zur Zugriffsbeschränkung zu nehmen und sicherzustellen, dass es bei der Beschaffung solcher Systeme eine entsprechende Auswahl gibt. Eine Lösung besteht darin, den einheitlichen Scrambling-Algorithmus zu verwenden und für den Zugriff Lösungen zu benutzen, die auf kommerziellen Abmachungen zwischen den Betreibern beruhen. Diese Lösung kann mit einzelnen CA-Systemen betrieben werden, die in die Decoder eingebaut sind.

Eine zweite Lösung beruht auf einer genormten Schnittstelle zwischen einem Modul und einem Hauptgerät, wobei in dem Modul Funktionen zur Zugriffsbeschränkung und mehr allgemein definierte, systemgebundene Funktionen implementiert sein können. Diese Lösung erlaubt auch den Rundfunkbetreibern Module zu benutzen, die in demselben Rundfunksystem Lösungen von verschiedenen Lieferanten enthalten, wodurch sich die Auswahl und die Möglichkeiten gegen Piraterie verbessern. Zweck dieses Schriftstückes ist es, diese Einheitliche Schnittstelle zu beschreiben.

Der Decoder, in dieser Spezifikation als Hauptgerät bezeichnet, enthält die erforderlichen Funktionen, um MPEG-2-Bild, -Ton und -Daten in Klartext zu empfangen. Diese Spezifikation legt die Schnittstelle zwischen dem Hauptgerät und den Scrambling- und CA-Anwendungen fest, die mit einem externen Modul betrieben werden.

Es werden zwei logische Schnittstellen definiert, die in derselben physikalischen Schnittstelle enthalten sind. Die erste Schnittstelle ist die Schnittstelle für den MPEG-2-Transportstrom. Die Verbindungsschicht und die physikalische Schicht werden in dieser Spezifikation, die höheren Schichten in den MPEG-2-Spezifikationen definiert. Die zweite Schnittstelle, die Befehlsschnittstelle, überträgt die Befehle zwischen dem Hauptgerät und dem Modul. Für diese Schnittstelle werden sechs Ebenen definiert. Ein Beispiel eines einzelnen Moduls in Verbindung mit einem Hauptgerät wird in Bild 1 gezeigt.

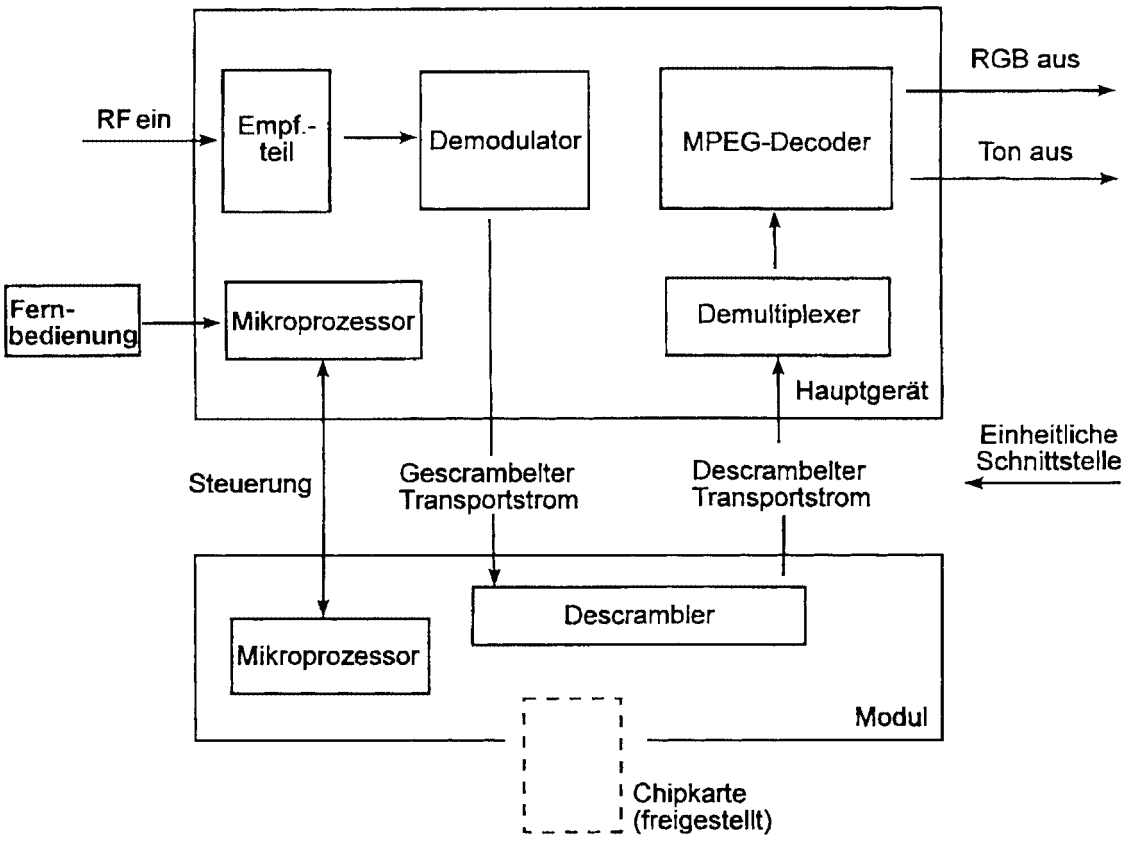

Bild 1 – Beispiel eines einzelnen Moduls in Verbindung mit dem Hauptgerät

Diese Spezifikation definiert nur solche Aspekte des Hauptgerätes, die erforderlich sind, um die Dialoge über diese Schnittstelle vollständig festzulegen. Die Spezifikation setzt nichts über die Konstruktion des Hauptgerätes voraus, mit der Ausnahme, dass eine Reihe von Hilfsmittel definiert wird, die das Hauptgerät benötigt, um das Modul betreiben zu können.

Die Spezifikation definiert nicht den Betrieb oder funktionelle Fähigkeiten einer CA-System-Anwendung im Modul. Die Anwendungen, die von einem Modul durchgeführt werden können, das über die Schnittstelle mit dem Hauptgerät in Verbindung steht, sind nicht auf Zugriffsbeschränkung oder auf Anwendungen begrenzt, die in dieser Spezifikation beschrieben sind. Es dürfen mehr als ein Modul gleichzeitig unterstützt werden.

2 Normative Verweisungen

Diese Europäische Norm enthält durch datierte oder undatierte Verweise Angaben aus anderen Publikationen. Diese normativen Verweisungen werden an den entsprechenden Stellen im Text zitiert und die Publikationen nachstehend aufgelistet. Für datierte Verweisungen gelten spätere Änderungen oder Überarbeitungen dieser Publikationen für die vorliegende Europäische Norm nur dann, wenn sie durch Änderung oder Überarbeitung aufgenommen sind. Für undatierte Verweisungen gilt die letzte Ausgabe der Publikation, auf die verwiesen wurde (einschließlich Änderungen).

[1] ISO/IEC 13818-1, *Generic Coding of Moving Pictures and Associated Audio: Systems.*[1)]

[2] ISO 8824:1987, *Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN. 1).*

[3] ISO 8825:1987, *Open Systems Interconnection – Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1).*

[4] ETS 300468, *Specification for Service Information (SI) in Digital Video Broadcasting (DVB) systems.*

[5] ETR 162, *Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) Systems.*

[6] PC Card Standard, *Volume 2 – Electrical Specification, February 1995, Personal Computer Memory Card International Association, Sunnyvale, California.*

[7] PC Card Standard, *Volume 3 – Physical Specification, February 1995, Personal Computer Memory Card International Association, Sunnyvale, California.*

[8] PC Card Standard, *Volume 4 – Metaformat Specification, February 1995, Personal Computer Memory Card International Association, Sunnyvale, California.*

[9] prETS 300743, *DVB Subtitling Specification.*

3 Definitionen

Für die Anwendung dieser Norm gelten die folgenden Definitionen:

Anwendung:
Eine Anwendung läuft in einem Modul, das mit dem Hauptgerät in Verbindung steht. Sie bietet dem Benutzer darüber hinaus solche Möglichkeiten, die direkt vom Hauptgerät zur Verfügung gestellt werden. Eine Anwendung kann den Transportstrom verarbeiten.

Hauptgerät:
Gerät, an das ein oder mehrere Module angeschlossen werden können, z. B. IRD, VCR, PC.

Modul:
Kleines Gerät, das nicht für sich alleine betrieben werden kann, hergestellt um zusammen mit dem Hauptgerät spezialisierte Aufgaben durchzuführen, zum Beispiel: ein Untersystem für Zugriffsbeschränkung, ein Anwendungsmodul für einen elektronischen Programmführer oder um Hilfsmittel bereitzustellen, die von einer Anwendung angefordert werden, aber von dem Hauptgerät nicht direkt geliefert werden können.

Hilfsmittel:
Funktionelle Einheit, die vom Hauptgerät zur Benutzung in einem Modul zur Verfügung gestellt wird. Ein Hilfsmittel definiert eine Reihe von Objekten, die zwischen einem Modul und dem Hauptgerät ausgetauscht werden, durch das das Modul das Hilfsmittel benutzt.

---

1) *Nationale Fußnote: Zu ergänzen ist ISO/IEC 13818-9; Information technology – Generic Coding of Moving Pictures and Associated Audio Information – Part 9: Extension for real time interface for real time decoders*

Dienst:
Reihe von Elementarströmen, die dem Benutzer als Programm angeboten werden. Sie sind durch eine gemeinsame Synchronisierung verbunden. Sie können aus verschiedenen Daten bestehen, d. h. Bild, Ton, Untertitel und andere Daten.

Transportstrom:
MPEG-2-Transportstrom.

4 Designphilosophie

4.1 Schichtung

Die Spezifikation wird in Schichten beschrieben, um zukünftige Varianten beim Implementieren aufnehmen zu können. Die Anwendungs- und Sitzungsschicht werden für alle Anwendungen der einheitlichen Schnittstelle definiert. Die Transport- und die Verbindungsschicht können von der physikalischen Schicht abhängig sein, die bei einer speziellen Implementierung benutzt wird. Die physikalische Schnittstelle wird innerhalb dieser Spezifikation definiert und umfasst die vollständige physikalische Spezifikation des Moduls.

Die Schichtung der Spezifikation ermöglicht Flexibilität beim Einsatz der Schnittstelle für eine Reihe von Anwendungen über CA hinaus. Die Schichtung lässt auch zu, dass es mehrere CA-Verfahren für dasselbe Hauptgerät gibt.

Die grundlegende Schichtung an der Befehlsschnittstelle ist in Bild 2 dargestellt. Das Hauptgerät darf Transportverbindungen mit mehr als einem Modul einrichten, die direkt oder indirekt mit dem Hauptgerät verbunden sind. Jede Verbindung bleibt so lange erhalten, wie das Modul vorhanden ist. Jedes Modul darf eine Anzahl unterschiedlicher Sitzungen mit dem Hauptgerät durchführen.

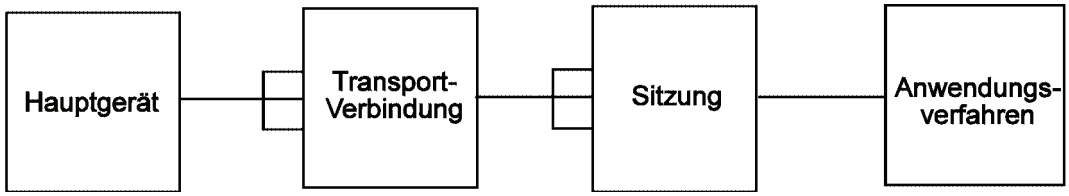

Bild 2 – Schichtung an der Befehlsschnittstelle

4.2 Physikalische Implementierung

Die Basisspezifikation enthält die Implementierung an einer physikalischen Schnittstelle, die mit der PC-Kartennorm kompatibel ist, die in der PC-Industrie benutzt wird. Andere physikalische Implementierungen sind in der Zukunft möglich.

4.3 Client-Server

Die Schnittstelle ist nach dem Prinzip konzipiert, dass Anwendungen, wie Clients, die von einem Server bereitgestellten Hilfsmittel benutzen. Die Anwendungen befinden sich in einem Modul und die Hilfsmittel können entweder durch das Hauptgerät oder ein anderes Modul auf eine von dem Hauptgerät bestimmte Art gehandhabt werden. Im englischen Text wurde für Hilfsmittel anstelle der Benennung 'services' vorzugsweise die Benennung 'resources' benutzt, weil 'services' beim Rundfunk allgemein für 'TV and radio services' verwendet wird und dort Unklarheiten vermieden werden sollten.

4.4 Codierung der Daten

Die Übertragung der Daten über die Befehlsschnittstelle wird in Form von Objekten definiert. Die Objekte werden mit Hilfe einer allgemeinen Tag-Längenwert(TLV)-Codierung codiert, die von der Syntax abgeleitet ist, die für das Codieren von ASN.1-Syntax benutzt wird (siehe [2] und [3]). Sie ist im Allgemeinen erweiterbar. Es gibt eine spezielle Transportschicht-Codierung für die PC-Karten-Implementierung, sie kann aber bei anderen physikalischen Implementierungen anders sein. Die Semantik ist jedoch identisch.

4.5 Erweiterbarkeit

Die höheren Schichten wurden so konzipiert, dass sie erweiterbar sind. Wie oben gezeigt, ist die benutzte TLV-Codierung erweiterbar, so dass neue Objekte hinzugefügt und bestehende Objekte erweitert werden können. Es gibt bezüglich des Platzbedarfs für die Tag-Codierung oder einer Längenbeschränkung des Wertes keine Probleme. Das Hilfsmittel „Hilfsmittelmanager“ bietet einen Mechanismus zur Erweiterung des Bereiches der Hilfsmittel, die von den Hauptgeräten sowohl für CA-Zwecke als auch für andere auf Modulen basierende Anwendungen bereitgestellt werden.

4.6 Einbringen bestehender Normen

Bestehende Normen wurden, wo möglich und geeignet, als Bausteine für diese Spezifikation benutzt. Dies ergibt wichtige zeitliche Marktvorteile, da die gesamte Entwicklungsarbeit für diese Normen schon geleistet wurde. Es bringt auch Vorteile bei der Implementierung, weil Software und Hardware, die schon für bestehende Normen entwickelt wurden, hier mit möglichen Kostenvorteilen wiederverwendet werden können.

5 Beschreibung und Architektur

5.1 Überblick

Um die Stelle im Hauptgerät zu definieren, an der die Einheitliche Schnittstelle logischerweise angeordnet sein kann, wird für ein Hauptgerät eine teilweise logische Architektur vorausgesetzt. In anderer Hinsicht wurde die Freiheit für den Entwickler des Hauptgerätes so wenig wie möglich eingeschränkt. Bild 1 zeigt ein vereinfachtes Bild einer typischen Hauptgerätearchitektur und die Lage der Schnittstelle. Dabei ist zu beachten, dass es in einem Hauptgerät mehr als eine Schnittstelle geben kann.

Die Einheitliche Schnittstelle besteht aus zwei Teilen, der Transportstromschnittstelle und der Befehlsschnittstelle. Beide sind geschichtet, um die Gesamtkonstruktion der Schnittstelle und die Implementierung zu erleichtern. Die höheren Schichten sind für alle Implementierungen einheitlich. Für die unteren Schichten sind alternative Implementierungen möglich. Diese Spezifikation enthält eine, die auf der PC-Kartennorm basiert. In zukünftigen Versionen können aber andere enthalten sein.

5.2 Transportstromschnittstelle

Die Transportstromschnittstelle überträgt MPEG-2-Transportpakete in beide Richtungen. Wenn das Modul den Zugriff auf bestimmte Dienste im Transportstrom freigibt und diese Dienste vom Hauptgerät ausgewählt wurden, dann werden die Pakete, die diese Dienste übertragen, descrambelt zurückgegeben. Die anderen Pakete werden nicht verändert. An der Transportstromschnittstelle wird durch das Modul und eine zugeordnete Bedingungslogik der physikalischen Schicht unter den meisten Bedingungen eine konstante Verzögerung aufrechterhalten (siehe 5.4.2). Die Schichten der Transportstromschnittstelle werden in Bild 3 gezeigt. Die Transportschicht und alle höheren Schichten werden in der MPEG-2-Spezifikation ISO 13818 definiert.

Höhere Schichten
Transportschicht
PC-Karten-Verbindungsschicht
PC-Karten-physikalische Schicht

Bild 3 – Schichten der Transportstromschnittstelle

5.3 Befehlsschnittstelle

Die Befehlsschnittstelle überträgt jede Art von Kommunikation zwischen den in dem Modul laufenden Anwendungen und dem Hauptgerät. Um für die notwendige Funktionsfähigkeit zu sorgen, werden die Kommunikationsprotokolle an dieser Schnittstelle in verschiedenen Schichten definiert. Diese Funktionsfähigkeit enthält die Fähigkeit, an einem Hauptgerät mehrere Module zu unterstützen, die Fähigkeit komplexe Kombinationen von Transaktionen zwischen Modul und Hauptgerät zu unterstützen und eine erweiterbare Reihe von funktionellen Grundformen (Objekte), die dem Hauptgerät erlauben, Hilfsmittel an die Module zu liefern. Die Schichtung wird nachstehend in Bild 4 gezeigt.

Die in dieser Spezifikation beschriebene PC-Karten-Implementierung hat ihre eigene physikalische und Verbindungsschicht und auch ihre eigene Untere Transport-Unterschicht. Eine zukünftige unterschiedliche Implementierung unterscheidet sich wahrscheinlich in diesen Schichten und der Unterschied wird auf diese Schichten beschränkt sein. Die implementierungsspezifischen Merkmale der Unteren Transport-Unterschicht sind auf das Codieren beschränkt und auf spezielle Einzelheiten des Nachrichtenaustauschprotokolls. Die gemeinsame Obere Unterschicht definiert Kennzeichnung, Beginnen und Beenden von Transportschichtverbindungen. Die Sitzungs-, Hilfsmittel- und Anwendungsschichten sind für alle physikalischen Implementierungen gemeinsam.

So weit wie möglich wurde die Anwendungsschicht der Schnittstelle so ausgeführt, dass sie frei von spezieller Anwendungssemantik ist. Die Kommunikation findet in Form von Hilfsmitteln, wie Benutzer-Schnittstellen-Dialog, und als Kommunikation mit niedriger Datenrate statt, die das Hauptgerät den in einem Modul laufenden Anwendungen zur Verfügung stellt. Diese Strategie macht es sehr viel leichter, Module für andere Aufgaben als nur die Zugriffsbeschränkung zu beschaffen.

<table>
<tr><td colspan="4">Anwendung</td></tr>
<tr><td colspan="4">Hilfsmittel:</td></tr>
<tr><td>Benutzerschnittstelle</td><td>Kommunikation niedriger Datenrate</td><td>System</td><td>Freigestelle Erweiterungen</td></tr>
<tr><td colspan="4">Sitzungsschicht</td></tr>
<tr><td colspan="4">Generische Transport-Unterschicht</td></tr>
<tr><td colspan="4">PC-Karten-Transport-Unterschicht</td></tr>
<tr><td colspan="4">PC-Karten-Verbindungsschicht</td></tr>
<tr><td colspan="4">PC-Karten-physikalische Schicht</td></tr>
</table>

Bild 4 – Schichtung der Befehlsschnittstelle

5.4 Physikalische Anforderungen

5.4.1 Einführung

Dieser Abschnitt definiert die Anforderungen, die die physikalische Schicht erfüllen muss, um alle benötigten Funktionen durchführen zu können. Die folgenden Kennwerte der physikalischen Schicht werden hier nicht eingeschränkt, obwohl in der Spezifikation für jede benutzte physikalische Schicht die mechanische und elektrische Verbindung zwischen dem Hauptgerät und dem Modul, d. h. Typ und Größe der Buchse, Anzahl der Kontaktstifte, Spannungen, Impedanzen und Leistungsgrenzwerte festgelegt werden.

Anforderungen und Grenzwerte der folgenden Kennwerte der physikalischen Schicht werden hier definiert:

– logische Verbindungen für Transportstrom und Befehle;

– Datenraten;

– Verhalten beim Verbinden und Trennen;

– Initialisierung auf unterer Ebene;

– Verwendung von mehreren Modulen.

5.4.2 Logische Verbindungen für Daten und Befehle

Die physikalische Schicht muss unabhängige logische Verbindungen in beiden Richtungen für den Transportstrom und für die Befehle unterstützen.

Die Transportstromschnittstelle muss einen MPEG-2-Transportstrom annehmen, der aus einer Folge von Transportpaketen besteht, entweder zusammenhängend oder durch Nulldaten getrennt. Der zurückgesendete Transportstrom kann einige der eingehenden Transportpakete in descrambelter Form enthalten. Die Transportstromschnittstelle ist Gegenstand der folgenden Beschränkungen:

1 Wenn das Modul die Quelle eines Transportstromes ist, muss sein Ausgangssignal ISO/IEC 13818-9 entsprechen.

2 Jedes Ausgangspaket muss zusammenhängend sein, wenn die Quelle des Paketes oder des Eingangspaketes zusammenhängend ist.

3 Ein Modul muss eine konstante Verzögerung einführen, wenn es ein Eingangs-Transportpaket mit maximaler Abweichung der Verzögerung (tmdv) verarbeitet, die nach folgender Gleichung für jedes Byte gilt:

$\mathrm{tmdv}_{\max} = (n * \mathrm{TMCLKI}) + (2 * \mathrm{TMCLKO})$

und

$\mathrm{tmdv}_{\max} \leq 1$ Mikrosekunde, wenn $n = 0$.

Dabei ist:

tmdv Abweichung der Verzögerung des Moduls;

n Anzahl der Lücken in dem entsprechenden Eingangs-Transportpaket;

TMCLKI Eingangsdaten-Taktperiode;

TMCLKO Ausgangsdaten-Taktperiode.

* Eine 'Lücke' wird durch eine MCLKI-Anstiegflanke definiert, für die das MIVAL-Signal inaktiv ist. Eine 'Lücke' ist immer ein Byte breit. Mehrere 'Lücken' dürfen aufeinander folgen.

* Alle Hauptgeräte sollten zusammenhängende Transportpakete ausgeben.
* Hauptgeräte dürfen nicht zusammenhängende Transportpakete nur dann abgeben, wenn sie weniger als 3 Buchsen für die einheitliche Schnittstelle implementieren.
* Die Anzahl der 'Lücken innerhalb eines Paketes' können beträchtlich voneinander abweichen.

4 Ein der Befehlsschnittstelle entsprechendes Hauptgerät sollte konzipiert sein, Nm Module zu unterstützen. Nm ist die größere Zahl der durch das Hauptgerät implementierten Buchsen der Befehlsschnittstelle oder 16. Es sollte den Jitter, der sich von den Nm Modulen ergibt, plus dem Jitter in dem Eingangs-Transportstrom tolerieren. Der Worst-Case-Jitter kann entweder von dem eigenen Eingang des Hauptgerätes, dem Nm Module folgen oder einem Eingangsmodul mit ISO/IEC 13818-9 entsprechendem Ausgang, auf den $(Nm - 1)$ Module folgen, herrühren.

5 Alle Schnittstellen müssen Datenraten von mindestens $58\ \mathrm{Mbit/s}$ unterstützen, gemittelt über den Zeitabschnitt zwischen den Synchronisationsbytes aufeinander folgender Transportpakete.

6 Alle Schnittstellen müssen eine Mindest-Byteübertragungs-Taktperiode von $111\ \mathrm{ns}$ unterstützen.

Die Befehlsschnittstelle muss Befehle, wie sie in dem entsprechenden Teil dieser Spezifikation über die Transportschicht definiert sind, in beiden Richtungen übertragen. Die unterstützte Datenrate muss in jeder Richtung mindestens $3{,}5\ \mathrm{Mbit/s}$ betragen.

5.4.3 Verhalten beim Verbinden und Trennen

Die physikalische Schicht muss Verbinden und Trennen des Moduls jederzeit unterstützen, unabhängig davon ob das Hauptgerät eingeschaltet ist oder nicht. Verbinden oder Trennen darf keinen elektrischen Schaden, weder am Modul noch am Hauptgerät, verursachen und darf keine unerwünschte Änderung von gespeicherten nichtflüchtigen Daten in dem Modul verursachen. Wenn ein Modul nicht angeschlossen ist, muss die Transportstromschnittstelle das Modul überbrücken, und die Befehlsschnittstelle zu diesem Modul muss inaktiv sein. Zum Verbinden des Moduls muss das Hauptgerät eine Initialisierungssequenz auf unterer Ebene mit dem Modul einleiten. Dies wird durchgeführt, unabhängig davon, welches Verfahren zum Einrichten der Verbindung auf unterer Ebene von der speziellen physikalischen Schicht benutzt wird, und dann so eingerichtet, dass das Modul ein DVB-konformes Modul ist. Nach erfolgreichem Abschluss muss das Hauptgerät die Transportstromverbindung durch Einfügen des Moduls in den Transportstrompfad des Hauptgeräts einrichten. Es wird akzeptiert, dass während dieses Vorgangs einige Transportstromdaten verloren gehen. Gleichzeitig muss eine Transportschichtverbindung an der Befehlsschnittstelle eingerichtet werden, um zu ermöglichen, dass die Initialisierung der Anwendungsschicht stattfindet und die übliche Anwendungsschicht-Kommunikation fortgeführt wird.

Wenn die physikalische Schicht in anderen Anwendungen als in einer DVB-konformen Modulverbindung benutzt wird und wenn ein nichtkonformes Modul an das Hauptgerät angeschlossen ist, darf keine Beschädigung am Modul oder Hauptgerät verursacht werden. Das Hauptgerät darf keine Initialisierung wie bei einem DVB-konformen Modul durchführen. Wahlweise kann das Hauptgerät dem Benutzer anzeigen, dass ein nicht erkanntes Modul angeschlossen wurde.

Beim Trennen des Moduls muss das Hauptgerät das Modul aus dem Transportstromdatenpfad wegnehmen. Es ist akzeptabel, dass einige Transportstromdaten während dieses Vorgangs verloren gehen. Auch die Befehlsschnittstellenverbindung muss durch das Hauptgerät abgeschlossen werden.

5.4.4 Mehrere Module

Die Anwendungsschicht setzt in der Anzahl der Module, die gleichzeitig an das Hauptgerät angeschlossen werden können, keine Grenzen. Bei speziellen physikalischen Schichten und speziellen Hauptgerätekonstruktionen kann dies jedoch vorkommen. Die Spezifikation der physikalischen Schicht muss zulassen, dass dort mehrere Module gleichzeitig angeschlossen werden, wenn auch bei einem Hauptgeräte-Minimalkonzept nur eine Verbindung vorhanden sein kann. Idealerweise sollte die Spezifikation der physikalischen Schicht bei der Anzahl der Module keine strengen Grenzen setzen, außer wenn eine Grenze vorgeschrieben wird. Die muss dann bei nicht weniger als 15 Modulen festgelegt werden.

Wenn es Einrichtungen dafür gibt, mehr als ein Modul anzuschließen, muss, wie in Bild 5 gezeigt, die Verbindung der Transportstromschnittstellen nacheinander durch jedes einzelne Modul durchgeschleift werden. Das Hauptgerät muss davon getrennte und gleichzeitige Befehlsschnittstellen-Verbindungen zu jedem Modul aufrechterhalten, so dass Transaktionen zwischen Hauptgerät und Modul für jedes Modul unabhängig durchgeführt werden können. Wenn ein Modul ausgestöpselt wird, darf die Befehlsschnittstellen-Transportschichtverbindung zu keinem anderen Modul gestört oder beendet werden.

Wenn mehrere Module an einem Hauptgerät angeschlossen sind, sollte das Hauptgerät in der Lage sein, die für das Descrambeln von ausgewählten Diensten verwendbare Module auszuwählen.

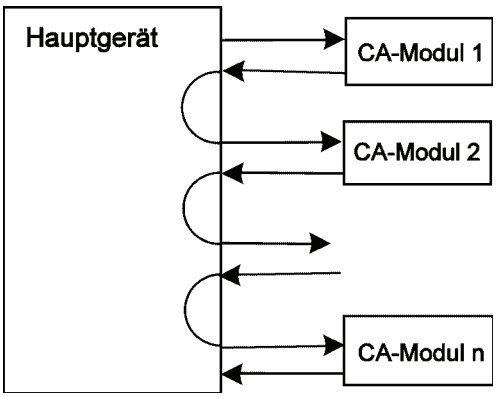

Bild 5 – Verkettung der Transportstromschnittstellen zwischen den Modulen

5.5 Betriebsbeispiel

Um einige der oben beschriebenen Merkmale zu verdeutlichen, behandelt dieses Beispiel die Vorgänge, die dann ablaufen, wenn ein PC-Kartenmodul in ein Hauptgerät gesteckt wird. Die Initialisierung der PC-Karte beginnt mit dem Abtasten des Moduls, das mit Abtaststiften an die Schnittstelle gesteckt wurde. Das Hauptgerät liest dann die Struktur der Karteninformation, die sich in dem Attributspeicher des Moduls befindet. Dieser enthält Konfigurationsinformationen auf unterer Ebene für das Modul, wie PC-Karten-Lesen und -Schreiben von Adressen, die von dem Modul benutzt werden, und zeigt dem Hauptgerät an, dass dies ein DVB-konformes Modul ist. Das Hauptgerät schaltet nun die Überbrückungsleitung der Transportstromschnittstelle ab und ermöglicht den Transportpaketen durch das Modul zu fließen. Dies führt zu einer Verzögerung und folglich zu einer kurzen Lücke in den Transportstromdaten, die jedoch unvermeidbar ist. Gleichzeitig findet der Initialisierungsprozess der physikalischen Schnittstelle statt, um die für die Kommunikation zu benutzende Puffergröße abzustimmen. An dieser Stelle ist der Initialisierungsvorgang der physikalischen Schicht abgeschlossen, und einheitlich für alle physikalischen Implementierungen beginnt der Initialisierungsvorgang der höheren Schicht damit, dass das Hauptgerät eine Transportschichtverbindung zu dem Modul herstellt. Dieser Vorgang und der übrige Initialisierungsvorgang der höheren Schicht werden an anderer Stelle in diesem Schriftstück beschrieben.

Der Initialisierungsvorgang wird logischerweise für die anderen physikalischen Implementierungen ähnlich sein, obwohl sich die Einzelheiten unterscheiden werden.

6 Transportstromschnittstelle (TSI)

6.1 TSI – physikalisch, Verbindungsschichten

Diese Schichten sind von der physikalischen Implementierung des Moduls abhängig.

6.2 TSI – Transportschicht

Die verwendete Transportschicht ist dieselbe wie die Transportschicht des MPEG-2-Systems. Die Datenbewegung über die Transportstromschnittstelle wird mit MPEG-2-Transportpaketen abgewickelt. Das gesamte MPEG-2-Multiplexsignal wird über diese Transportstromschnittstelle übertragen und voll oder teilweise descrambelt wieder empfangen. Wenn das Paket nicht gescrambelt ist, gibt das Modul es so wie es ist zurück. Wenn es gescrambelt ist und das Paket gehört zu dem ausgewählten Dienst und das Modul kann den Zugriff auf diesen Dienst freigeben, dann gibt das Modul das entsprechend descrambelte Paket mit auf '00' gesetztem transport_scrambling_control-Flag zurück.

Wenn das Scrambeln auf der Ebene des Paketierten Elementarstromes (PES) durchgeführt wird, dann reagiert das Modul in derselben Weise und unter gleichen Bedingungen wie oben, und gibt den entsprechend descrambelten PES mit dem auf '00' gesetzten PES_scrambling_control-Flag zurück.

Das Transportpaket und das PES-Paket werden in der MPEG-2-Systemspezifikation [1] vollständig definiert.

6.3 TSI – obere Schichten

Bis auf den Paketierten Elementarstrom ist jede Schichtung oder Struktur der MPEG-2-Daten oberhalb der Transportstromschicht für diese Spezifikation nicht relevant. Die Spezifikation setzt jedoch voraus, dass das Modul bestimmte Daten, die für seinen Betrieb benötigt werden, wie ECMs (Berechtigungs-Steuerungsmeldungen) und EMMs (Berechtigungs-Verwaltungsmeldungen), findet und direkt aus dem Transportstrom entnimmt.

7 Befehlsschnittstelle – Transport- und Sitzungsschichten

Die Kommunikation der Daten über die Befehlsschnittstelle wird in Form von Objekten definiert. Die Objekte werden mit Hilfe einer allgemeinen Tag-Längenwert-Codierung codiert, die von der Codierung der ASN.1-Syntax abgeleitet ist.

Tabelle 1 – Längenfeld, das bei allen Protokolldateneinheiten bei Transport-, Sitzungs- und Anwendungsschichten benutzt wird

Syntax	Anzahl der Bits	Mnemonik
length_field() {		
size-indicator	**1**	**bslbf**
if (size_indicator == 0)		
length_value	**7**	**uimsbf**
else if (size_indicator == 1) {		
length_field_size	**7**	**uimsbf**
for (i=0; i<length_field_size; i++){		
length_value_byte	**8**	**bslbf**
}		
}		
}		

Dieser Abschnitt beschreibt die ASN.1-Objekte für die Transport- und die Sitzungsschichten, die über die Befehlsschnittstelle laufen. Bei all diesen Objekten und den Objekten der Anwendungsschicht in Abschnitt 8 gilt für das Längenfeld, das die Anzahl der Bytes in dem darauf folgenden Wertfeld anzeigt, die Codierung nach Tabelle 1.

Das erste Bit des length_field ist der size_indicator. Wenn der size_indicator = 0, wird die Länge des Datenfeldes in den nachfolgenden 7 Bits codiert. Somit kann jede Länge von 0 bis 127 in einem Byte codiert werden. Wenn die Länge 127 übersteigt, wird der size_indicator auf 1 gesetzt. Dann werden in den nachfolgenden 7 Bits die Anzahl der in dem Längenfeld nachfolgenden Bytes codiert. Um einen ganzzahligen Wert zu codieren, müssen diese nachfolgenden Bytes verkettet werden, das erste Byte mit dem höchstwertigen Ende. So kann jede Wertfeldlänge bis zu 65535 mit drei Bytes codiert werden.

Das unbestimmte Längenformat, das durch die Basiscodierregeln von ASN.1 (siehe [3]) festgelegt ist, wird nicht benutzt.

7.1 Generische Transportschicht

7.1.1 Einführung

Die Transportschicht der Befehlsschnittstelle arbeitet oberhalb einer Verbindungsschicht, die durch die benutzte spezielle physikalische Implementierung zur Verfügung gestellt wird. Für die physikalische Implementierung der Basis-PC-Karte wird die Verbindungsschicht in Anhang A beschrieben. Das Transportprotokoll setzt voraus, dass die Verbindungsschicht zuverlässig ist, d. h. dass die Daten in richtiger Reihenfolge und ohne Löschen und Wiederholen transportiert werden.

Das Transportprotokoll ist ein Befehl-Antwort-Protokoll, wobei das Hauptgerät an das Modul einen Befehl sendet, und dabei eine Befehls-Transportprotokoll-Dateneinheit (C_TPDU) benutzt und auf eine Antwort von dem Modul mit einer Antwort-Transportprotokoll-Dateneinheit (R_TPDU) wartet. Das Modul kann die Kommunikation nicht einleiten: es muss auf das Hauptgerät warten, das es abfragt oder zuerst Daten sendet. Das Protokoll wird durch elf Transportschichtobjekte unterstützt. Einige davon treten nur in C_TPDUs vom Hauptgerät auf, einige nur in R_TPDUs vom Modul und einige können in beiden auftreten. Create_T_C und C_T_C_Reply erstellen eine neue Transportverbindungen. Delete_T_C und D_T_C_Reply löschen sie. Mit Request_T_C und New_T_C kann ein Modul das Hauptgerät auffordern, eine neue Transportverbindung herzustellen. Mit T_C_Error können Fehlerbedingungen signalisiert werden. T_SB überträgt Statusinformation vom Modul zum Hauptgerät. T_RCV fordert von einem Modul abrufbereite Daten an und T_Data_More und T_Data_Last befördern Daten von höheren Schichten zwischen Hauptgerät und Modul. T_Data_Last mit einem leeren Datenfeld wird vom Hauptgerät benutzt, um vom Modul regelmäßig Daten abzufragen, wenn es selbst keine Daten zu senden hat.

Ein C_TPDU vom Hauptgerät enthält nur ein Transportprotokollobjekt. Ein R_TPDU vom Modul kann ein oder zwei Transportprotokollobjekte übertragen. Das einzige oder das zweite Objekt eines Paars in einer R_TPDU ist immer ein Objekt T_SB.

7.1.2 Transportprotokollobjekte

Alle Transportschichtobjekte enthalten eine Transportverbindungskennung. Diese ist ein Oktett, das ermöglicht, dass bis zu 255 Transportschichtverbindungen an dem Hauptgerät gleichzeitig aktiv sind. Transportverbindungs-Kennungswert 0 ist reserviert. Der Kennungswert wird immer von dem Hauptgerät zugewiesen. Das Protokoll wird hier in Einzelheiten beschrieben, da es für alle physikalischen Implementierungen einheitlich ist. Die Objekte werden aber nur allgemein beschrieben. Die detaillierte Codierung der Objekte hängt von der benutzten speziellen physikalischen Schicht ab. Die Codierung für die physikalische Implementierung der PC-Karte wird in dem normativen Anhang A beschrieben.

Am Hauptgerät muss es möglich sein, mindestens 16 Transportverbindungen je unterstützter Modulbuchse zu erstellen, wobei aber vorzugsweise alle 255 Verbindungen unter den Modulbuchsen verteilt sein sollten.

1 Create_T_C erstellt die Transportverbindung und wird nur vom Hauptgerät ausgegeben und überträgt den Transportverbindungs-Kennungswert für die einzurichtende Verbindung.

2 C_T_C_Reply ist die Antwort vom Zielmodul auf Create_T_C und überträgt die Transportverbindungskennung für die erstellte Verbindung.

3 Delete_T_C löscht eine bestehende Transportverbindung und hat als Parameter die Transportverbindungskennung für die zu löschende Verbindung. Das Objekt kann sowohl vom Hauptgerät als auch vom Modul ausgegeben werden. Wenn es vom Modul ausgegeben wird, ist es als Antwort auf Abfragen oder auf Daten des Hauptgerätes.

4 D_T_C_Reply ist die Antwort auf das Löschen. Unter manchen Umständen kann diese Antwort ihr Ziel nicht erreichen, da das Objekt Delete_T_C eine ihm zugeordnete Zeitbegrenzung für die Antwort hat. Wenn diese Zeitgrenze erreicht ist, bevor die Antwort empfangen wird, dann können alle Aktionen, die beim Empfang der Antwort durchgeführt werden sollten, zu dieser Zeitgrenze durchgeführt werden.

5 Request_T_C fordert das Hauptgerät auf, eine neue Transportverbindung zu erstellen. Das Objekt wird über eine bestehende Transportverbindung vom Modul gesendet. Es wird als Antwort auf Abfragen oder auf Daten des Hauptgerätes gesendet.

6 New_T_C ist die Antwort auf Request_T_C. Sie wird auf derselben Transportverbindung gesendet wie das Objekt Request_T_C und überträgt die Transportverbindungskennung für die neue Verbindung. Auf New_T_C folgt unmittelbar ein Objekt Create_T_C für die neue Verbindung, das die Transportverbindung entsprechend einrichtet.

7 T_C_Error wird gesendet, um einen Fehler zu signalisieren und überträgt einen 1-Byte-Fehlercode, der den Fehler beschreibt. In dieser Version wird T_C_Error nur als Antwort auf Request_T_C gesendet, um zu signalisieren, dass keine weiteren Transportverbindungen zur Verfügung stehen.

8 T_SB wird als Antwort auf alle vom Hauptgerät empfangenen Objekte gesendet, entweder an andere Protokollobjekte angehängt oder gegebenenfalls selbständig. T_SB überträgt ein Byte, das anzeigt, wenn beim Modul Daten zum Absenden vorhanden sind.

9 T_RCV wird vom Hauptgerät gesendet, um anzufordern, dass die Daten, die im Modul zum Senden vorliegen (in einer vorangegangenen T_SB vom Modul signalisiert), zum Hauptgerät zurückgesendet werden.

10 T_Data_More und T_Data_Last befördern Daten zwischen Hauptgerät und Modul und können entweder in einer C_TPDU oder einer R_TPDU sein. Sie werden von dem Modul immer nur als Antwort auf eine ausdrückliche Anforderung des Hauptgeräts durch ein T_RCV gesendet. T_Data_More wird benutzt, wenn eine Protokolldateneinheit (PDU) von einer höheren Schicht in Teilstücke aufgespalten werden muss, um sie bei einer externen Beschränkung der Größe der Datenübertragung zu senden. Das Objekt zeigt an, dass mindestens noch ein Teilstück der PDU der oberen Schicht nach diesem einen gesendet wird. T_Data_Last zeigt das letzte oder einzige Teilstück einer PDU der oberen Schicht an.

7.1.3 Transportprotokoll

Die Bilder 6 und 7 zeigen die Zustandsübergangsdiagramme für das Einrichten und Löschen einer Verbindung auf der Seite des Hauptgeräts bzw. des Moduls. Jeder Zustandsübergangsbogen wird mit dem Ereignis gekennzeichnet, das diesen Übergang verursacht. Wenn der Übergang auch ein zu sendendes Objekt verursacht, dann wird dies mit einem eingerahmten Text angegeben.

Wenn das Hauptgerät eine Transportverbindung zu einem Modul einrichten will, sendet es das Objekt Create_T_C und geht in den Zustand 'im Aufbau'. Das Modul muss unmittelbar mit einem C_T_C_Reply-Objekt antworten. Wenn nach einer begrenzten Zeit das Modul nicht antwortet, kehrt das Hauptgerät in den Zustand 'nicht in Betrieb' – über den Bogen 'Zeitbegrenzung' – zurück. Bei dieser speziellen Transportverbindung wird

das Hauptgerät nicht noch einmal senden oder abfragen. Ein zu spätes C_T_C_Reply wird ignoriert. Wenn das Hauptgerät dieselbe Transportverbindungskennung wieder benutzt, wird das Modul wieder Create_T_C empfangen und daraus muss es ableiten, dass die alte Transportverbindung außer Betrieb ist und eine neue eingerichtet werden muss.

Wenn das Modul mit C_T_C_Reply antwortet, geht das Hauptgerät in den Zustand 'Aktiv' der Verbindung. Wenn das Hauptgerät Daten zu senden hat, kann es sie nun senden, sonst gibt es aber eine Abfrage aus und fragt dann danach regelmäßig die Verbindung ab.

Wenn das Hauptgerät die Transportverbindung beenden will, sendet es ein Objekt Delete_T_C und geht in den Zustand 'in Löschung'. Es geht dann bis zum Empfang eines Objektes D_T_C_Reply oder nach einer Zeitbegrenzung, während der nichts empfangen wird, in den Zustand 'nicht in Betrieb' zurück. Wenn das Hauptgerät ein Objekt Delete_T_C vom Modul empfängt, gibt es ein Objekt D_T_C_Reply aus und geht direkt in den Zustand 'nicht in Betrieb'. Außer in dem Zustand 'Aktiv' wird jedes Objekt ignoriert, das in einem Zustand empfangen wird, der nicht erwartet wurde.

In dem Zustand 'Aktiv' sendet das Hauptgerät periodisch Abfragen, oder es sendet Daten, wenn es eine PDU der oberen Schicht senden muss. Als Antwort empfängt es ein Objekt T_SB, eingeleitet durch ein Objekt Request_T_C bzw. Delete_T_C.

In dem Zustand 'Aktiv' können jederzeit vom Hauptgerät Daten gesendet werden. Das Modul kann erst Daten senden, wenn es eine Nachricht vom Hauptgerät empfangen hat – üblicherweise Daten oder eine Abfrage – und dann in der Antwort T_SB anzeigen, dass Daten vorhanden sind. Das Hauptgerät wird dann zu einem Zeitpunkt – nicht notwendigerweise sofort – eine Anforderung T_RCV an das Modul senden, auf die das Modul durch Senden eines Objektes T_Data mit den abrufbereiten Daten antwortet. Wenn T_Data_More benutzt wird, muss jedes darauf folgende Teilstück auf eine weitere Anforderung T_RCV vom Hauptgerät warten, bevor es gesendet werden kann.

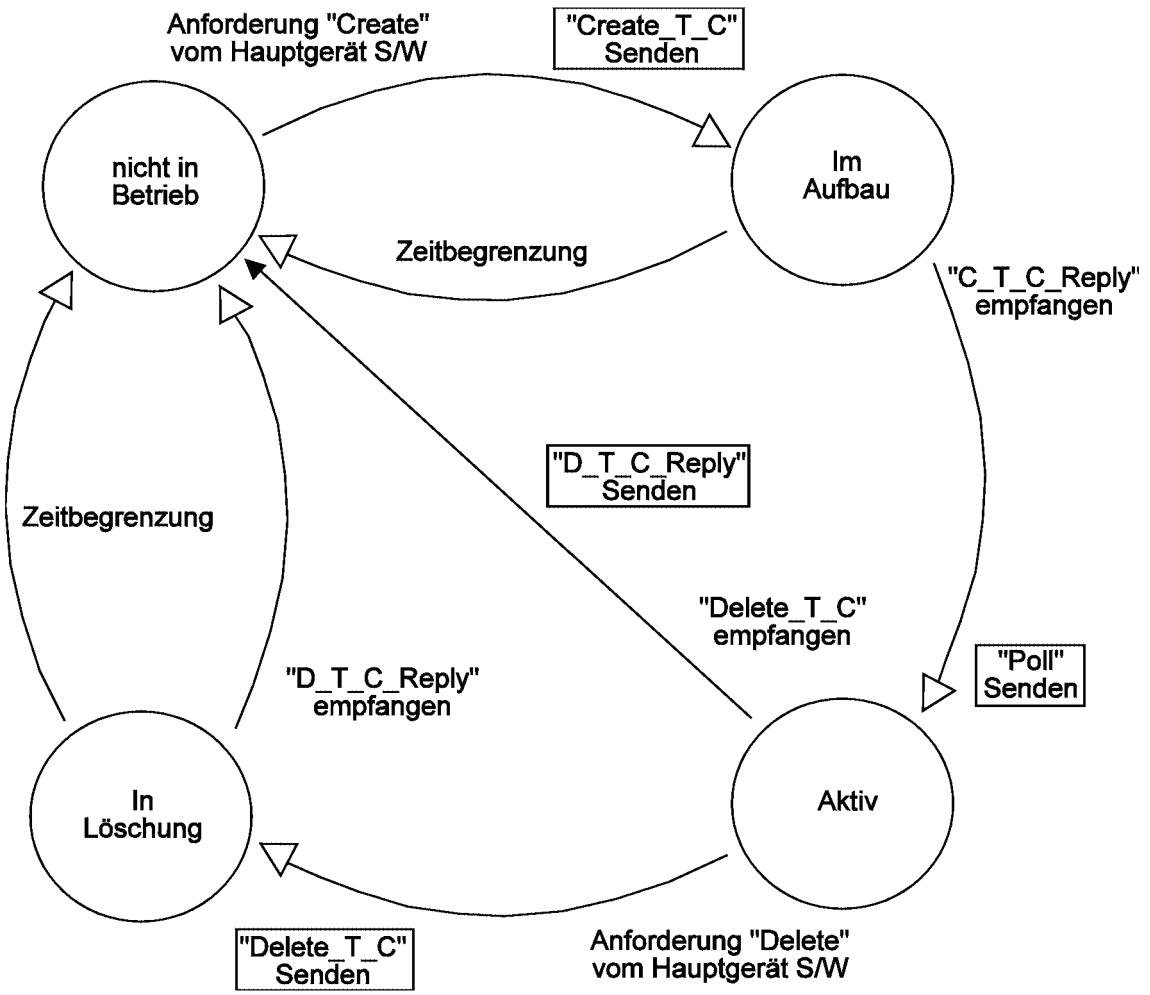

Bild 6 – Diagramm der Zustandsübergänge für die Hauptgerätseite des Transportprotokolls

Tabelle 2 – Objekte, deren Empfang bei den Zuständen einer Transportverbindung am Hauptgerät erwartet wird

Zustand	Erwartete Objekte – Hauptgerät
Nicht in Betrieb	Keine
Im Aufbau	C_T_C_Reply (+ T_SB)
Aktiv	T_Data_More, T_Data_Last, Request_T_C, Delete_T_C, T_SB
In Löschung	D_T_C_Reply (+ T_SB)

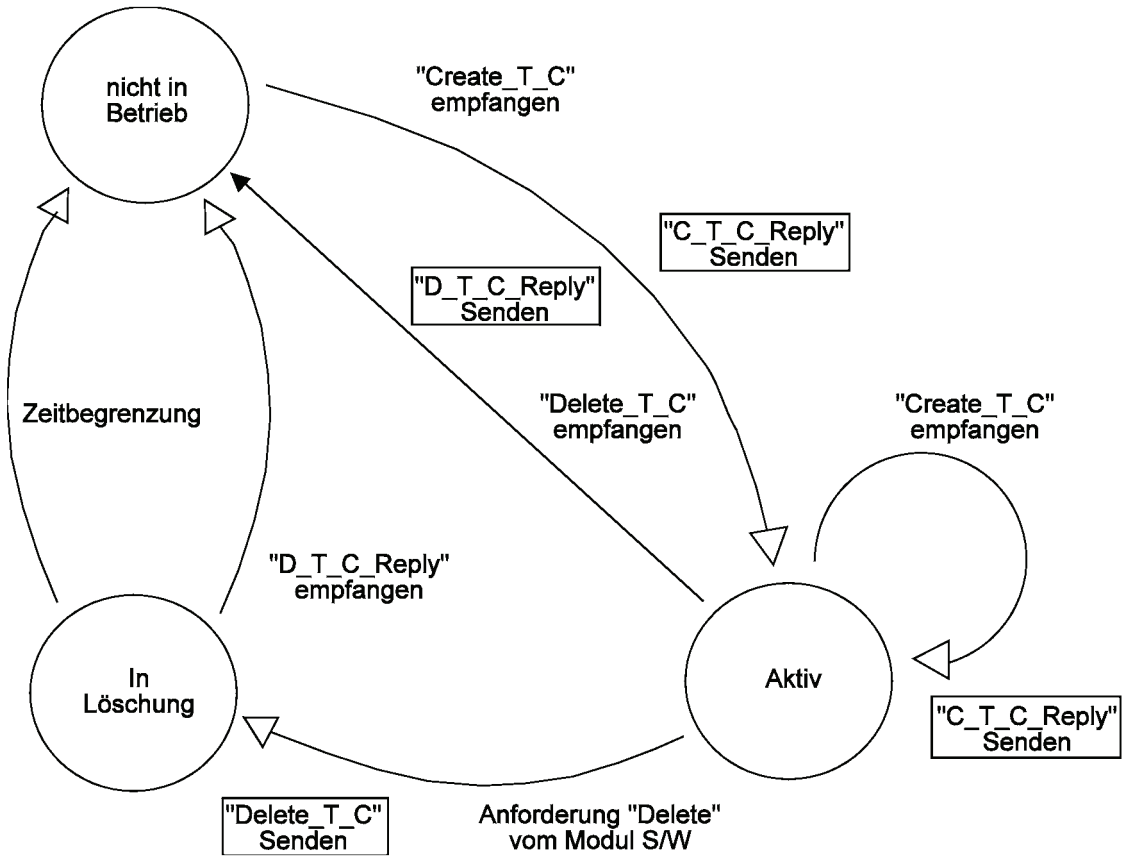

Bild 7 – Diagramm der Zustandsübergänge für die Modulseite des Transportprotokolls

Tabelle 3 – Objekte, deren Empfang bei den Zuständen einer Transportverbindung am Modul erwartet wird

Zustand	Erwartete Objekte – Modul
Nicht in Betrieb	Create_T_C_
Aktiv	Create_T_C, T_Data_More, T_Data_Last, New_T_C, Delete_T_C, T_RCV, T_C_Error
In Löschung	D_T_C_Reply

Zum Einrichten einer weiteren Transportverbindung muss das Modul das Objekt Request_T_C entweder als Antwort auf eine Abfrage oder auf Daten senden. Wenn das Hauptgerät die Anforderung erfüllen kann, wird es mit einer New_T_C antworten, die die Transportverbindungskennung für die neue Verbindung enthält, unmittelbar gefolgt von einem Objekt Create_T_C, um die Verbindung zu erstellen. Wenn das Hauptgerät die Anforderung nicht erfüllen kann, weil alle Transportverbindungskennungen in Benutzung sind, dann muss es mit T_C_Error antworten, das den entsprechenden Fehlercode enthält.

Ein Beispiel einer Objektübertragungsfolge für das Erstellen und Benutzen einer Transportverbindung wird in Bild 8 gezeigt.

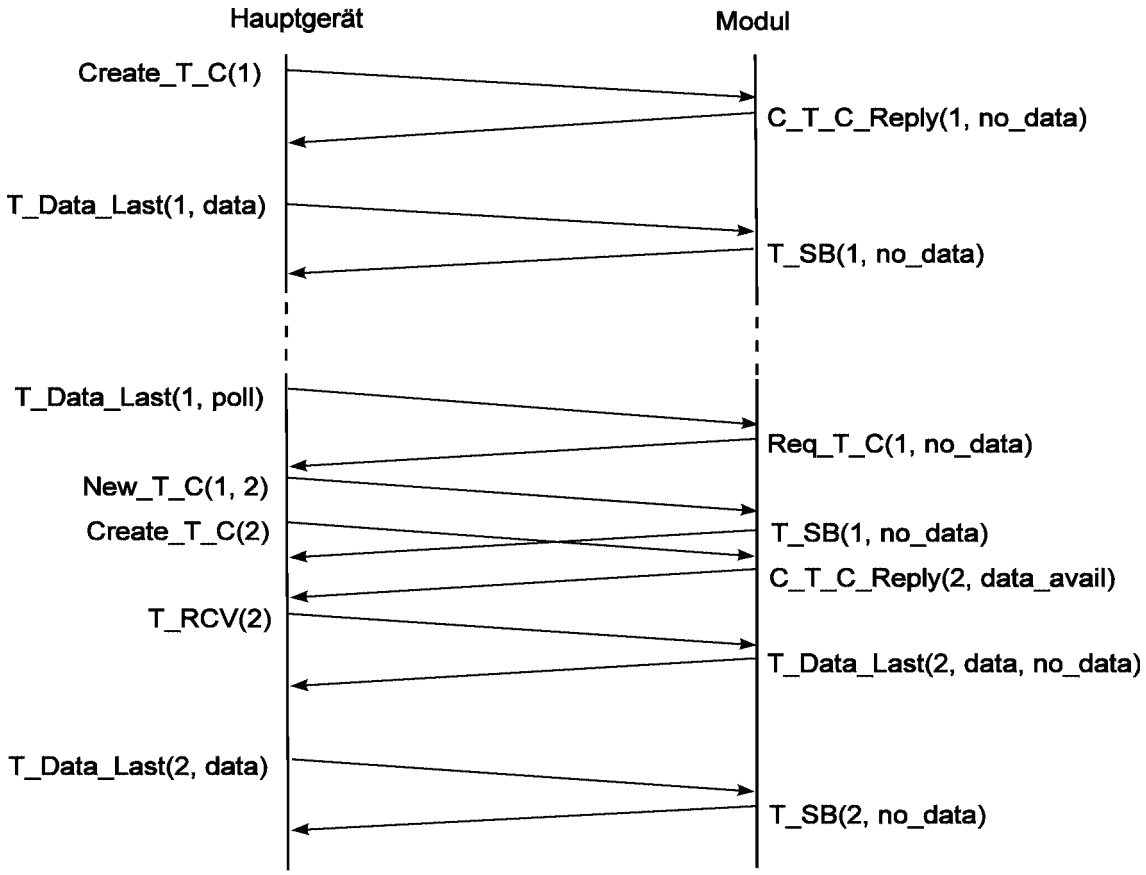

Bild 8 – Beispiel einer Objektübertragungsfolge für das Transportprotokoll

In diesem Beispiel wird angenommen, dass das Modul gerade eingesteckt und eine physikalische Verbindung (PC-Karten-Initialisierung usw.) eingerichtet wurde. Das Hauptgerät gibt nun ein Create_T_C für die Transportverbindung Nummer 1 aus (wenn schon andere Module eingesteckt sind, ist dies eine höhere Nummer). Das Modul antwortet unmittelbar mit C_T_C_Reply für die Transportverbindung 1 und zeigt auch an, dass es keine Daten zu senden hat. Das Hauptgerät sendet einige Daten mit einem T_Data_Last und das Modul antwortet nur mit einem T_SB und zeigt damit an, dass es keine Daten zu senden hat. Einige Zeit später fragt das Hauptgerät das Modul mit einem leeren T_Data_Last ab. Das Modul antwortet mit einem Request_T_C was bedeutet, dass es eine neue Transportverbindung einrichten muss, und zeigt auch an (in dem angehängten T_SB), dass es keine Daten hat, um sie über Verbindung 1 zu senden. Das Hauptgerät antwortet mit New_T_C und zeigt damit an, dass die Transportverbindung 2 eingerichtet wird. Unmittelbar darauf folgt Create_T_C für Transportverbindung 2. Das Modul antwortet mit einem T_SB auf die erste und einem C_T_C_Reply auf die zweite Verbindung und zeigt damit auch an, dass es über diese Verbindung Daten zu senden hat. Das Hauptgerät antwortet mit T_RCV, um die Daten zu empfangen, und das Modul antwortet mit T_Data_Last, das die Daten enthält. Das Hauptgerät antwortet mit seinen eigenen Daten und das Modul antwortet darauf, dass er keine weiteren Daten zu senden hat. Beide Transportverbindungen bleiben nun bestehen, bis entweder das Modul oder das Hauptgerät eine davon löscht. Das Hauptgerät fragt regelmäßig auf beiden Verbindungen mit einem leeren T_Data_Last ab.

7.2 Sitzungsschicht

7.2.1 Einführung

Die Sitzungsschicht stellt den Mechanismus zur Verfügung, durch den die Anwendungen mit den Hilfsmitteln kommunizieren und von den Hilfsmitteln Gebrauch machen. Das Hilfsmittel ist ein Mechanismus zum Zusammenfassen von funktionellen Fähigkeiten in der Anwendungsschicht und wird in 8.2 vollständig beschrieben.

Hilfsmittel unterscheiden sich in der Anzahl der gleichzeitigen Sitzungen, die sie unterstützen können. Manche Hilfsmittel unterstützen nur eine Sitzung. Wenn eine zweite Anwendung versucht, eine Sitzung mit einem Hilfsmittel anzufordern, das schon in Gebrauch ist, wird sie eine Antwort 'Hilfsmittel belegt' empfangen. Andere Hilfsmittel können mehr als eine Sitzung gleichzeitig unterstützen, wobei dann Hilfsmittelanforderungen bis zu einer von dem Hilfsmittel festgelegten Grenze berücksichtigt werden. Ein Beispiel für diesen Fall ist das Hilfsmittel 'Anzeigen', das bei einigen Hauptgeräteimplementierungen in der Lage ist, gleichzeitige Anzeigen in verschiedenen Fenstern zu unterstützen.

7.2.2 Sitzungsprotokollobjekte

Die Sitzungsobjekte werden hier so allgemein ausgedrückt beschrieben, wie es dem Protokoll entspricht. Die ausführliche Codierung der Objekte wird in späteren Abschnitten beschrieben.

– open_session_request wird von einer Anwendung über ihre Transportverbindung an das Hauptgerät ausgegeben, das die Benutzung eines Hilfsmittels anfordert. Das Hauptgerät kann das Hilfsmittel direkt unterstützen oder es kann von einem anderen Modul (über eine zweite Transportverbindung) unterstützt werden, wobei das Hauptgerät das Objekt create_session benutzt, um die Sitzung über die geeignete Transportverbindung zu erweitern.

– open_session_response wird von dem Hauptgerät an eine Anwendung geantwortet, die ein Hilfsmittel angefordert hat, um eine Sitzungsnummer zuzuteilen, oder um dem Modul mitzuteilen, dass seine Anforderung nicht erfüllt werden kann.

– create_session wird von dem Hauptgerät an einen Hilfsmittelversorger in einem Modul ausgegeben, um eine Sitzungsanforderung von einer Anwendung auf eine andere Transportverbindung zu erweitern.

– create_session_response wird von dem Hilfsmittelversorger in einem Modul an das Hauptgerät geantwortet, so dass das Hauptgerät dem Ausgangsmodul mitteilen kann, ob die Sitzung eröffnet werden kann.

– close_session_request wird von einem Modul oder dem Hauptgerät ausgegeben, um eine Sitzung zu schließen.

– close_session_response wird von einem Modul oder dem Hauptgerät ausgegeben, um das Schließen der Sitzung zu bestätigen.

– session_number leitet immer den Inhalt der SPDU ein, die APDU(s) enthält.

7.2.3 Sitzungsprotokoll

Der Dialog in der Sitzungsschicht wird durch ein Modul oder das Hauptgerät eingeleitet. Nachstehend werden zwei Beispiele erläutert. Im ersten Beispiel (Bild 9) benutzt ein Modul A ein Hilfsmittel, das vom Hauptgerät zur Verfügung gestellt wird. In dem zweiten Beispiel (Bild 10) fordert ein Modul A an, ein Hilfsmittel zu benutzen, das vom Modul B zur Verfügung gestellt wird.

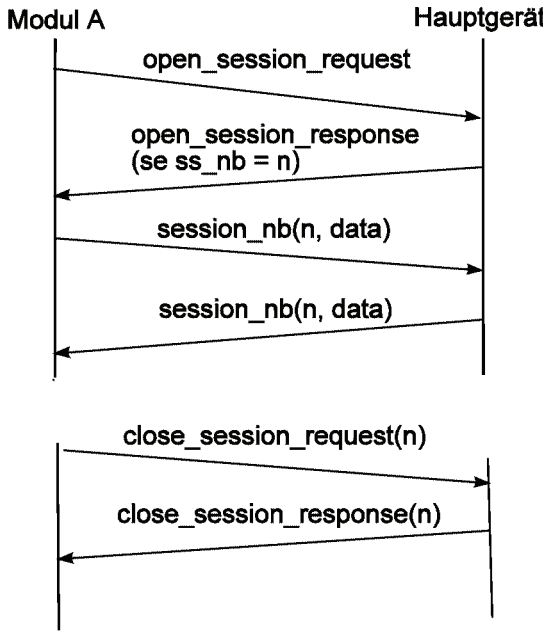

Bild 9 – Sitzung mit einem vom Hauptgerät gelieferten Hilfsmittel

Modul A fordert für ein Hilfsmittel auf seiner Transportverbindung eine Sitzung zu eröffnen. Da das Hauptgerät selbst das Hilfsmittel liefert, antwortet es in seiner 'Sitzung-Eröffnen-Antwort' direkt mit einer Sitzungsnummer. Die Kommunikation wird nun mit Anwendungsschichtdaten fortgeführt, die durch Objekte session_number eingeleitet werden. Schließlich wird die Sitzung geschlossen, in diesem Beispiel durch das Modul A. Sie könnte aber auch durch das Hauptgerät geschlossen werden, z. B. wenn das Hilfsmittel aus irgendeinem Grund nicht mehr verfügbar ist.

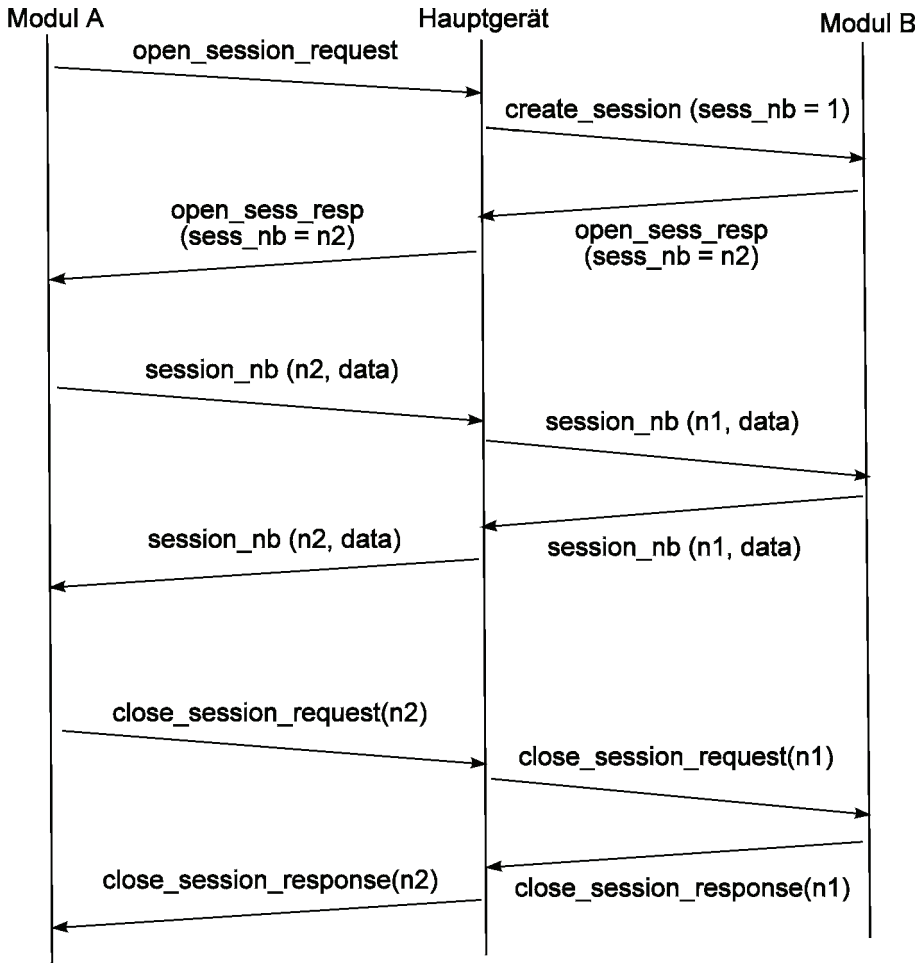

Bild 10 – Sitzung mit einem vom Modul gelieferten Hilfsmittel

Modul A fordert für ein Hilfsmittel, auf seiner Transportverbindung eine Sitzung zu öffnen. Das Hauptgerät kann dieses Hilfsmittel von dem Modul B erhalten, so dass das Hauptgerät mit einem Objekt create_session die Sitzung auf die Transportverbindung des Hilfsmittelversorgers erweitert. Der Hilfsmittelversorger antwortet darauf, und das Hauptgerät antwortet dann wieder dem ursprünglich anfordernden Modul A. Die Kommunikation der Anwendungsschichtdaten wird fortgeführt und das Hauptgerät führt die notwendige Lenkung zwischen eingehenden und ausgehenden Transportverbindungen durch. Zuletzt wird die Sitzung geschlossen, in diesem Beispiel durch Modul A. Sie könnte aber auch durch das Hauptgerät oder das Modul B geschlossen werden. Die Sitzungsnummern n1 und n2 werden vom Hauptgerät zugeteilt. In diesem Beispiel werden sie als unterschiedlich angegeben. Bei der Implementierung des Hauptgerätes kann gewählt werden, ob es bei den einzelnen Transportverbindungen dieselben oder unterschiedliche Sitzungsnummern benutzt. Die Anwendungsschicht kann keine Voraussetzungen dafür schaffen, dass am Ende der Sitzung die Sitzungsnummern dieselben sind oder nicht.

7.2.4 SPDU-Struktur

Die Sitzungsschicht benutzt zum Austausch der Daten auf Sitzungsebene entweder vom Hauptgerät zum Modul oder vom Modul zum Hauptgerät die Struktur der Sitzungsprotokoll-Dateneinheit (SPDU).

Die SPDU-Struktur wird nachstehend in Bild 11 und Tabelle 4 gezeigt:

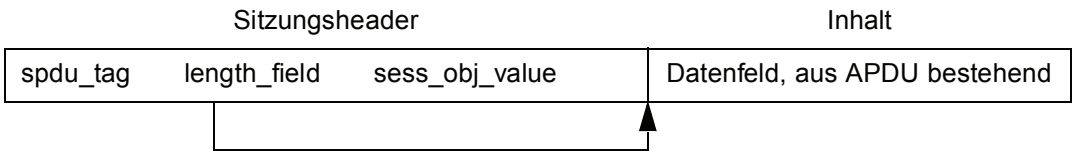

Bild 11 – SPDU-Struktur

Tabelle 4 – Codierung der SPDU

Syntax	Anzahl der Bits	Mnemonik
SPDU() {		
spdu_tag	**8**	**uimsbf**
length_field()		
for (i=0;i<length_value;i++)		
session_object_value byte	**8**	**uimsbf**
}		
for (i=0;i<N;i+-+) {		
} apdu()		
}		

Die SPDU besteht aus folgenden zwei Teilen:

– einem **obligatorischen** Sitzungsheader, bestehend aus einem Tag-Wert spdu_tag, einem length_field, das die Länge des Sitzungsobjekt-Wertfeldes codiert, und einem Sitzungsobjektwert. Diese Sitzungsobjekte werden nachstehend in 7.2.6 beschrieben. Hinweis: Das Längenfeld enthält nicht die Länge von gegebenenfalls nachfolgenden APDUs;

– einem bedingten Hauptteil variabler Länge, der eine ganzzahlige Anzahl von APDUs enthält, die zu derselben Sitzung gehören (siehe Anwendungsschicht). Das Vorhandensein des Hauptteils hängt von dem Sitzungsheader ab.

7.2.5 Transport der SPDU

Eine SPDU wird in dem Datenfeld von einer oder mehreren TPDUs transportiert. Für weitere Information siehe die Beschreibung der TPDU der einzelnen physikalischen Modulimplementierung.

7.2.6 Beschreibung der Sitzungsheader

Die nachstehend aufgelisteten Objekte sind Sitzungsheaderobjekte. Auf nur einen SPDU-Header folgt ein Datenfeld – das Objekt 'session_number' – dem immer ein SPDU-Hauptteil folgt, der eine oder mehrere APDUs enthält.

7.2.6.1 Anforderung Sitzung Öffnen (open_session_request)

Dieses Objekt wird von dem Modul an das Hauptgerät ausgegeben, um das Öffnen einer Sitzung zwischen dem Modul und einem Hilfsmittel anzufordern, das entweder vom Hauptgerät oder vom Modul zur Verfügung gestellt wird. Die resource_identifier muss sowohl nach Klasse als auch nach Typ zu einem Hilfsmittel passen, das in der Liste der verfügbaren Hilfsmittel des Hauptgerätes vorkommt. Wenn das Versionsfeld der gelieferten Hilfsmittelkennung 0 ist, dann benutzt das Hauptgerät die aktuelle Version in seiner Liste. Wenn die Versionsnummer in der Anforderung niedriger oder gleich der Versionsnummer in der Liste des Hauptgerätes ist, dann wird die aktuelle Version benutzt. Wenn die angeforderte Versionsnummer höher als die Version in der Liste des Hauptgerätes ist, dann lehnt das Hauptgerät die Anforderung mit dem entsprechenden Antwortcode ab.

Tabelle 5 – Codierung der Anforderung Sitzung Öffnen

Syntax	Anzahl der Bits	Mnemonik
open_session_request () {		
open_session_request_tag	**8**	**uimsbf**
length_field() = 4		
resource_identifier() /* siehe 8.2.2. */		
}		

7.2.6.2 Sitzung-Öffnen-Antwort (open_session_response)

Dieses Objekt wird vom Hauptgerät an das Modul ausgegeben, um eine Sitzungsnummer zuzuteilen oder dem Modul mitzuteilen, dass seine Anforderung nicht erfüllt werden kann.

Tabelle 6 – Codierung der Sitzung-Öffnen-Antwort

Syntax	Anzahl der Bits	Mnemonik
open_session_response() {		
open_session_response_tag	**8**	**uimsbf**
length_field() = 7		
session_status	**8**	**uimsbf**
resource_identifier()		
session_nb	**16**	**uimsbf**
}		

session_status values (Sitzungsstatuswerte)

Tabelle 7 – Statuswerte von Sitzung Öffnen

session status	Sitzungsstatus	Sitzungsstatuswert (hex)
session is opened	Sitzung ist eröffnet	00
session not opened, resource non-existent	Sitzung nicht eröffnet, Hilfsmittel nicht vorhanden	F0
session not opened, resource exists but unavailable	Sitzung nicht eröffnet, Hilfsmittel vorhanden, jedoch nicht verfügbar	F1
session not opened, resource exists but version lower then requested	Sitzung nicht eröffnet, Hilfsmittel vorhanden, jedoch niedrigere Version als angefordert	F2
session not opened, resource busy	Sitzung nicht eröffnet, Hilfsmittel belegt	F3
other	Andere	Reserviert

resource_identifier (Hilfsmittelkennung)

Das Hauptgerät antwortet mit der aktuellen resource_identifier des angeforderten Hilfsmittels zusammen mit der aktuellen Versionsnummer. Wenn die Antwort 'resource non-existent' ist, dann wird das resource_identifier-Feld mit dem Feld identisch sein, das in der Anforderung Sitzung Öffnen geliefert wurde.

session_nb (Sitzungsnummer)

Vom Hauptgerät zugeteilte Nummer für die angeforderte Sitzung. Wert 0 ist reserviert. Die session_nb wird für den gesamten nachfolgenden Datenaustausch von APDUs zwischen dem Modul und dem Hilfsmittel (über das Hauptgerät) benutzt, bis die Sitzung geschlossen wird. Wenn die Sitzung nicht geöffnet werden kann (session_status ungleich 0), hat der session_value keine Bedeutung.

7.2.6.3 Sitzung Erstellen (create_session)

Dieses Objekt wird vom Hauptgerät an ein Modul ausgegeben, das ein Hilfsmittel liefert, um das Öffnen einer Sitzung anzufordern.

Tabelle 8 – Codierung von Sitzung Erstellen

Syntax	Anzahl der Bits	Mnemonik
create_session () {		
create_session_tag	**8**	**uimsbf**
length_field() = 6		
resource_identifier()		
session_nb	**16**	**uimsbf**
}		

resource_identifier (Hilfsmittelkennung)
Ist der Hilfsmittelkennungswert des angeforderten Hilfsmittels, vollständig mit der aktuellen Versionsnummer.

session_nb (Sitzungsnummer)
Ist die Sitzungsnummer, die vom Hauptgerät für die neue Sitzung zugeteilt wurde.

7.2.6.4 Sitzung-Erstellen-Antwort (create_session_response)
Dieses Objekt wird von dem Modul ausgegeben, das dem Hauptgerät ein Hilfsmittel liefert, um dem Modul mitzuteilen, ob die Sitzung eröffnet werden kann.

Tabelle 9 – Codierung der Sitzung-Erstellen-Antwort

Syntax	Anzahl der Bits	Mnemonik
create_session_response () {		
create_session_response_tag	**8**	**uimsbf**
length_field() = 7		
session_status	**8**	**uimsbf**
resource_identifier()		
session_nb	**16**	**uimsbf**
}		

session_status values (Sitzungsstatuswerte)
Es wird derselbe Statuswert benutzt wie für open_session_response – siehe Tabelle 7.

resource_identifier (Hilfsmittelkennung)
Das Modul setzt in die create_session_response die Klasse und den Typ des angeforderten Hilfsmittels ein, jedoch mit der Versionsnummer, die das Modul zur Zeit unterstützt.

session_nb (Sitzungsnummer)
Hat denselben Wert wie das session_nb-Feld in dem Objekt create_session, für das dies eine Antwort ist.

7.2.6.5 Anforderung Sitzung Schließen (close_session_request)
Dieses Objekt wird vom Modul oder vom Hauptgerät ausgegeben, um eine Sitzung zu schließen.

Tabelle 10 – Codierung der Anforderung Sitzung Schließen

Syntax	Anzahl der Bits	Mnemonik
close_session_request () {		
close_session_request_tag	**8**	**uimsbf**
length_field() = 2		
session_nb	**16**	**uimsbf**
}		

7.2.6.6 Sitzung-Schließen-Antwort (close_session_response)
Dieses Objekt wird vom Modul oder vom Hauptgerät ausgegeben, um das Schließen der Sitzung zu bestätigen.

Tabelle 11 – Codierung der Sitzung-Schließen-Antwort

Syntax	Anzahl der Bits	Mnemonik
create_session_response () {		
close_session_response_tag	**8**	**uimsbf**
length_field() = 3		
session_status	**8**	**uimsbf**
session_nb	**16**	**uimsbf**
}		

session_status values (Sitzungsstatuswerte)

Tabelle 12 – Statuswerte für Sitzung Schließen

Sitzungsstatus	Sitzungsstatuswert (hex)
Sitzung ist wie gefordert geschlossen	00
Sitzungsnummer in der Anforderung ist nicht zugeteilt	F0
Andere	Reserviert

7.2.6.7 Sitzungsnummer (session_number)

Dieses Objekt geht immer einem Hauptteil der SPDU voraus, die APDU(s) enthält.

Tabelle 13 – Codierung der Sitzungsnummer

Syntax	Anzahl der Bits	Mnemonik
session_number () {		
session_number_tag	**8**	**uimsbf**
length_field() = 2		
session_nb	**16**	**uimsbf**
}		

7.2.7 Codierung der Sitzungs-Tags

Die nachstehende Tabelle 14 enthält die Namen der Objekte, die von der Sitzungsschicht an der Befehlsschnittstelle benutzt werden. Die Codierung des spdu_tag folgt den ASN.1-Regeln. Jedes spdu_tag wird in einem Byte codiert.

Tabelle 14 – Sitzungs-Tag-Werte

spdu-Tag	Tag-Wert (hex)	Einfach (P) oder zusammengesetzt	Richtung Hauptgerät ↔ Modul
$T_{open\_session\_request}$	'91'	P	←
$T_{open\_session\_response}$	'92'	P	→
$T_{create\_session}$	'93'	P	→
$T_{create\_session\_response}$	'94'	P	←
$T_{close\_session\_request}$	'95'	P	↔
$T_{close\_session\_response}$	'96'	P	↔
$T_{session\_number}$	'90'	P	↔

Weitere Werte sind in den Bereichen 80..8F, 90..9F, A0..AF, B0..BF reserviert.

8 Befehlsschnittstelle – Anwendungsschicht

8.1 Einführung

Die Anwendungsschicht implementiert eine Reihe von Protokollen, die auf dem Konzept eines Hilfsmittels basieren. Ein Hilfsmittel definiert eine funktionelle Einheit, die den in einem Modul laufenden Anwendungen zur Verfügung steht. Jedes Hilfsmittel unterstützt eine Reihe von Objekten und ein Protokoll, um zur Benutzung des Hilfsmittels die Objekte auszutauschen. Kommunikation mit einem Hilfsmittel wird mit einer Sitzung durchgeführt, die von diesem speziellen Hilfsmittel erstellt wurde. Dieser Abschnitt beschreibt den Mindestsatz von Hilfsmitteln, die alle mit dieser Spezifikation konformen Hauptgeräte bereitstellen müssen. Es ist zu beachten, dass einige Hilfsmittel in Abschnitt 8 als Vorsilbe in ihrem Namen „DVB“ haben. Diese Hilfsmittel machen Gebrauch von DVB-spezifischen Merkmalen, die nicht in allen möglichen Situationen vorhanden sein können, in denen diese Schnittstelle benutzt werden kann. Die Hilfsmittel mit der Vorsilbe DVB müssen in allen DVB-konformen Hauptgeräten zur Verfügung stehen und dürfen wahlweise von anderen Hauptgeräten geliefert werden. Weitere wahlfreie Hilfsmittel werden in Anhang B zu dieser Spezifikation beschrieben und dürfen in weiteren Anhängen beschrieben werden.

8.2 Hilfsmittel

8.2.1 Einführung

Ein Hilfsmittel darf von dem Hauptgerät direkt zur Verfügung gestellt werden oder es ist in einem anderen Modul gespeichert. Ein Hilfsmittel wird durch die Hilfsmittelkennung gekennzeichnet, die drei Komponenten enthält: Hilfsmittelklasse, Hilfsmitteltyp und Hilfsmittelversion. Die Hilfsmittelklasse definiert eine Reihe von Objekten und ein Protokoll zu deren Benutzung. Der Hilfsmitteltyp definiert verschiedene Hilfsmitteleinheiten innerhalb einer Klasse. Alle Hilfsmitteltypen innerhalb einer Klasse benutzen dieselben Objekte und dasselbe Protokoll, bieten jedoch unterschiedliche Funktion oder sind unterschiedliche Fälle derselben Funktion. Die Hilfsmittelversion ermöglicht dem Hauptgerät, die letzte Version eines Hilfsmittels zu erkennen (höchste Versionsnummer), wenn mehr als ein Hilfsmittel derselben Klasse und Typ vorhanden sind. Dies ermöglicht aktualisierte oder erweiterte Hilfsmittel, die an ein Modul geliefert werden müssen, um im Hauptgerät oder einem anderen Modul vorhandene Hilfsmittel zu ersetzen. Hilfsmittel mit höherer Versionsnummer müssen zu vorhergehenden Versionen rückwärtskompatibel sein, so dass Anwendungen, die eine vorhergehende Version anfordern, ein Hilfsmittel mit dem erwarteten Verhalten bekommen.

Hilfsmittel werden von einer Anwendung benutzt, die eine Sitzung mit einem Hilfsmittel (siehe 7.2) erstellt. Durch einen vom Hilfsmittel-Manager durchgeführten Initialisierungsprozess hat das Hauptgerät alle verfügbaren Hilfsmittel identifiziert, unabhängig davon, ob das Hauptgerät sie selbst oder ob sie ein anderes Modul bereitstellt, und es kann die Sitzung an der entsprechenden Stelle abschließen. Wenn die Sitzung erstellt wurde, kann die Anwendung das Hilfsmittel durch Austausch von Objekten entsprechend dem definierten Protokoll benutzen.

Ein Beispiel für die Anwendung des Hilfsmittelkonzeptes ist die Hilfsmittelklasse Kommunikation niedriger Datenrate (8.7.1). Diese stellt einen allgemeinen Mechanismus zur Benutzung einer seriellen Kommunikationsverbindung zur Verfügung, typischerweise ein Modem oder ein Rückkanal in einem Kabelsystem. Für verschiedene Übertragungsgeschwindigkeiten des Modems werden Hilfsmitteltypen definiert, die durch die Nummern der ITU-T-Empfehlungen z. B. V.21, V.22, V.32bis gekennzeichnet sind. Die meisten modernen Modems bieten einen Bereich von Geschwindigkeiten, so dass ein Modem mehrere Hilfsmitteltypen enthalten kann. Die Wahl der Übertragungsgeschwindigkeit wird durch Erstellen einer Sitzung mit dem speziellen Hilfsmitteltyp durchgeführt, den das Modem anbietet.

8.2.2 Hilfsmittelkennung (resource_identifier)

Eine Hilfsmittelkennung besteht aus vier Achtbitzeichen. Die beiden höchstwertigen Bits des ersten Achtbitzeichens zeigen an, ob das Hilfsmittel öffentlich oder privat ist, und somit die Struktur des übrigen Feldes. Werte von 0,1 und 2 zeigen ein öffentliches Hilfsmittel an, ein Wert 3 ein privates Hilfsmittel.

Öffentliche Hilfsmittelklassen haben Werte im Bereich von 1 bis 49 150, die das resource_id_type-Feld als den höchstwertigen Teil der resource_class behandeln. Der Wert 0 ist reserviert. Der Maximalwert (ein und derselbe) aller Felder ist reserviert. Private Hilfsmittel werden durch einen privaten Hilfsmitteldefinierer festgelegt, das ist die Organisation, die ein privates Hilfsmittel definiert. Jeder private Hilfsmitteldefinierer kann die Struktur und den Inhalt des private_resource_identify-Feldes auf jede von ihm gewählte Art definieren, mit der Ausnahme, dass der Maximalwert (ein und derselbe) reserviert ist.

Tabelle 15 – Codierung der Hilfsmittelkennung

Syntax	Anzahl der Bits	Mnemonik
resource_identifier() {		
resource_id_type	**2**	**uimsbf**
if (resource_id_type != 3) {		
resource_class	**14**	**uimsbf**
resource_type	**10**	**uimsbf**
resource_version	**6**	**uimsbf**
}		
else {		
private_resource definer	**10**	**uimsbf**
private_resource_identity	**20**	
}		**uimsbf**
}		

8.2.3 Anwendungen und Hilfsmittelversorger

Dies sind die beiden Funktionsgruppen der Anwendungsschicht, die in einem Modul vorkommen können. Anwendungen machen von Hilfsmitteln Gebrauch, um für den Benutzer des Hauptgerätes Aufgaben zu lösen. Hilfsmittelversorger liefern Hilfsmittel zusätzlich zu denen, die im Hauptgerät direkt verfügbar sind, oder eine neuere Version eines Hilfsmittels, die eine vorher vom Hauptgerät oder einem anderen Modul gelieferte Version ersetzt. Um Probleme mit Blockierungen und Schwierigkeiten bei der Initialisierung zu vermeiden, dürfen die Hilfsmittelversorger nicht von dem Vorhandensein irgendwelcher anderer Hilfsmittel abhängig sein, mit Ausnahme des Hilfsmittelmanagers, der die Hilfsmittel liefert, die sie anbieten.

8.3 Anwendungsprotokoll-Dateneinheit (APDU)

8.3.1 Einführung

Alle Protokolle in der Anwendungsschicht benutzen zum Senden der Anwendungsdaten zwischen Modul und Hauptgerät oder zwischen den Modulen eine einheitliche Struktur für die Anwendungsprotokoll-Dateneinheit (APDU).

Die APDU-Struktur wird nachstehend in Bild 12 und Tabelle 16 dargestellt:

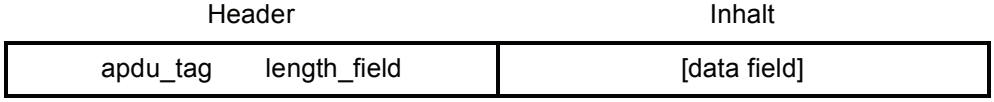

Bild 12 – APDU-Struktur

Tabelle 16 – Codierung der APDU

Syntax	Anzahl der Bits	Mnemonik
APDU() {		
apdu_tag	**24**	**uimsbf**
length_field()		
for (i=0;i<length_value;i++) {		
data_byte	**8**	**uimsbf**
}		
}		

Die APDU besteht aus zwei Teilen:

– einem vorgeschriebenen Header, der den Tag-Wert apdu_tag enthält, der anzeigt, welcher Parameter innerhalb des Datenfeldes gesendet wird, und ein Längenwert, der mit der Länge des folgenden Datenfeldes codiert ist. apdu_tag und length_field sind entsprechend den Basiscodierungsregeln von ASN.1 codiert;

– einem bedingten Hauptteil variabler Länge gleich der durch length_field codierten Länge.

Die APDU-Hauptteile werden in den folgenden Abschnitten beschrieben. Diese Liste kann in zukünftigen Versionen erweitert werden.

8.3.2 Verketten von APDU-Datenfeldern

Das in der APDU befindliche Datenfeld darf, wenn es wegen der Größe der Übertragungs- und Empfangspuffer des Hauptgerätes und des Moduls erforderlich ist, in mehrere Blöcke geringerer Größe aufgeteilt werden. Dieser Mechanismus steht nur einigen APDUs zur Verfügung. Die Verkettung wird durch die Verwendung von zwei unterschiedlichen apdu_tag-Werten (M_apdu_tag und L_apdu_tag) durchgeführt. Alle Blöcke des Datenfeldes, außer dem letzten, werden innerhalb einer APDU mit einem M_apdu_tag-Wert gesendet. Dieser Tag-Wert zeigt an, dass noch mehr Daten in einer weiteren APDU gesendet werden. Der letzte Block des Datenfeldes wird innerhalb einer APDU mit einem L_apdu_tag-Wert gesendet. Wenn der letzte Block empfangen ist, verkettet die Empfangseinrichtung alle empfangenen Datenfelder.

Dieser Mechanismus ist für alle APDUs mit zwei definierten Werten (M_apdu_tag und L_apdu_tag) gültig. Er ist in nachstehendem Bild 13 dargestellt.

8.3.3 Transport von APDUs innerhalb von SPDU

Eine ganzzahlige Anzahl von APDUs, die zu derselben Sitzung gehört, kann in dem Inhalt einer SPDU befördert werden.

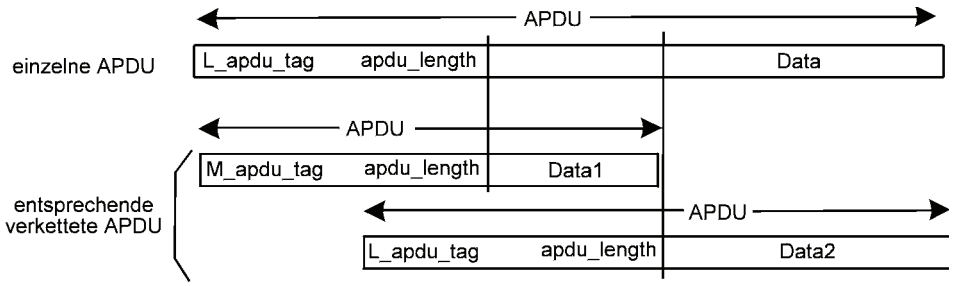

Bild 13 – Verkettungsmechanismus

8.4 Systemmanagement-Hilfsmittel

8.4.1 Hilfsmittelmanager

Der Hilfsmittelmanager ist ein vom Hauptgerät zur Verfügung gestelltes Hilfsmittel. Es gibt in der Klasse nur einen Typ und er kann jede Anzahl von Sitzungen unterstützen. Er steuert die Erfassung und Bereitstellung von Hilfsmitteln für alle Anwendungen. Zwischen Modul und Hauptgerät wird ein symmetrisches Kommunikationsprotokoll definiert, um die Hilfsmittel zu bestimmen, die jeder liefern kann. Das Protokoll wird zuerst vom Hauptgerät benutzt, um dann wieder jede Transportverbindung abzufragen und festzustellen, welche Hilfsmittel gegebenenfalls für die Benutzung in dieser Transportverbindung vorhanden sind. Dann wird es von Anwendungen benutzt, um alle verfügbaren Hilfsmittel zu ermitteln. Es wird dann periodisch benutzt, wenn Hilfsmittel sich ändern, um die allgemeine Übersicht der verfügbaren Hilfsmittel zu aktualisieren. Der Hilfsmittelmanager wird vom Hauptgerät zur Verfügung gestellt und kann nicht durch ein Hilfsmittel in einem Modul ersetzt werden. Jeder Versuch, ein Hilfsmittel 'Hilfsmittelmanager' durch ein Modul zu liefern, muss vom Hauptgerät ignoriert werden.

8.4.1.1 Hilfsmittelmanagerprotokoll

Wenn ein Modul eingesteckt oder das Hauptgerät eingeschaltet wird, werden ein oder vielleicht zwei Transportverbindungen zu dem Modul erstellt, die für eine Anwendung und/oder einen Hilfsmittelversorger bestimmt sind. Das erste, was eine Anwendung oder ein Hilfsmittelversorger ausführt, ist eine Sitzung für das Hilfsmittel 'Hilfsmittelmanager' anzufordern, die immer erstellt wird, weil der Hilfsmittelmanager keine Sitzungsbegrenzung hat. Der Hilfsmittelmanager sendet dann eine Profilanfrage an die Anwendung oder den Hilfsmittelversorger, die mit einer Profilantwort antworten, in der gegebenenfalls die Hilfsmittel aufgelistet sind, die sie bereitstellen. Die Anwendung oder der Hilfsmittelversorger müssen nun auf ein Objekt Profiländerung warten. Während des Wartens auf die Profiländerung kann sie (er) weder Sitzungen mit anderen Hilfsmitteln erstellen, noch kann sie (er) Sitzungen von anderen Anwendungen akzeptieren, die mit 'Hilfsmittel existiert nicht' bzw. 'Hilfsmittel existiert aber nicht verfügbar' antworten.

Wenn der Hilfsmittelmanager auf allen Transportverbindungen nach Profilen gefragt und Profilantworten empfangen hat, stellt das Hauptgerät eine Liste der verfügbaren Hilfsmittel zusammen. Wenn zwei oder mehr Hilfsmittel in beiden Klassen und Typen übereinstimmen, bleibt das Hauptgerät bei dem mit der höchsten Versionsnummer in seiner Liste. Wo die Versionsnummern auch übereinstimmen, behält das Hauptgerät alle Hilfsmittel und wählt eines davon zufällig, wenn eine Anforderung 'Sitzung Erstellen' dafür empfangen wird. Wenn das Hauptgerät seine Hilfsmittelliste aufgestellt hat, sendet es ein Objekt Profiländerung an alle augenblicklichen Hilfsmittelmanager-Sitzungen, und die Anwendungen können dann unter Benutzung des Objektes Profilanforderung beim Hauptgerät nach dessen Hilfsmittelliste anfragen.

Wenn sie (er) die Profiländerungs-Benachrichtigung das erste Mal empfängt, kann die Anwendung oder der Hilfsmittelversorger mit einer Profilanforderung beim Hauptgerät anfragen und eine Profilantwort mit der Liste der verfügbaren Hilfsmittel des Hauptgerätes empfangen. Nach diesem ersten Einsatz des Profiländerungs-Protokolls ist die Anwendung oder der Hilfsmittelversorger frei, um weitere Sitzungen zu erstellen oder zu akzeptieren. Ihre (seine) Sitzung mit dem Hilfsmittelmanager besteht weiter, um von Zeit zu Zeit weitere Profiländerungs-Benachrichtigungen durch das Hauptgerät zu ermöglichen.

Wenn ein Hilfsmittelversorger eine Änderung im Profil der Hilfsmittel, die er liefert, mitteilen will, gibt er eine Profiländerung an das Hauptgerät. Das Hauptgerät antwortet mit einer Profilanfrage, auf die der Hilfsmittelversorger dann wieder mit ihrer aktualisierten Hilfsmittelliste antwortet. Das Hauptgerät verarbeitet diese Antworten. Wenn sich daraus irgendeine Änderung für die eigene Hilfsmittelliste des Hauptgerätes ergibt, gibt das Hauptgerät eine Profiländerung an alle aktiven Hilfsmittelmanager-Sitzungen. Die Anwendungen können dann eine aktualisierte Hilfsmittelliste anfordern und empfangen.

8.4.1.2 Profilanfrage (profile_enq)

Das Objekt Profilanfrage fordert den Empfänger auf, mit einer Liste der Hilfsmittel zu antworten, die er in einem Objekt Profilantwort bereitstellt.

Tabelle 17 – Codierung des Objektes Profilanfrage

Syntax	Anzahl der Bits	Mnemonik
profile_enq () {		
profile_enq_tag	**24**	**uimsbf**
length_field()=0		
}		

8.4.1.3 Profilantwort (profile_reply)

Sie wird als Antwort auf eine Profilanfrage gesendet und listet die Hilfsmittel auf, die der Absender zur Verfügung stellen kann. Hilfsmittelkennungen für den Mindestsatz von Hilfsmittel, die bereitgestellt werden müssen, werden in 8.8 aufgelistet. Ferner können wahlweise Hilfsmittel, die definiert wurden, in Anhängen zu dieser Spezifikation aufgelistet werden.

Tabelle 18 – Codierung des Objektes Profilantwort

Syntax	Anzahl der Bits	Mnemonik
profile_reply () {		
profile_reply_tag	**24**	**uimsbf**
length_field() = N*4		
for (i=0; i<N; i++) {		
resource_identifier()		
}		
}		

8.4.1.4 Profil geändert (profile_changed)

Das Objekt Profil geändert teilt dem Empfänger mit, dass ein Hilfsmittel geändert wurde. Ein Modul würde es typischerweise benutzen, um dem Hauptgerät mitzuteilen, dass der Verfügbarkeitsstatus irgendeines seiner Hilfsmittel geändert wurde (nur dann nicht, wenn ein Hilfsmittel in Benutzung war). Das Hauptgerät würde gegebenenfalls seine eigene Hilfsmittelliste ändern und wenn eine Änderung erfolgte, würde es dann wieder an alle Transportverbindungen ein Objekt Profiländerung senden.

Tabelle 19 – Codierung des Objektes Profil geändert

Syntax	Anzahl der Bits	Mnemonik
profile_changed () {		
profile_changed_tag	**24**	**uimsbf**
length_field()=0		
}		

8.4.2 Anwendungsinformation

Das Hilfsmittel Anwendungsinformation macht es Anwendungen möglich, dem Hauptgerät einen Standardsatz von Informationen über sich selbst zu geben. Wie der Hilfsmittelmanager wird sie nur von dem Hauptgerät geliefert und hat keine Sitzungsbegrenzung. Alle Anwendungen erstellen zu diesem Hilfsmittel eine Sitzung, sobald sie ihre Profilanfragephase der Initialisierung abgeschlossen haben. Das Hauptgerät sendet dann ein Objekt 'Anfrage Anwendungsinformation' an die Anwendung, die durch Rücksenden eines Objektes 'Anwendungsinformation' mit der entsprechenden Information antwortet. Diese Sitzung wird aufrechterhalten, so dass das Hauptgerät jederzeit der Anwendung signalisieren kann, eine MMI-Sitzung an ihrer Einsprungstelle des Menüs auf höherer Ebene zu erstellen. Dies geschieht mit dem Objekt 'Menü Eingeben'. Wenn die Anwendung das Objekt 'Menü Eingeben' empfängt, muss sie eine MMI-Sitzung erstellen (siehe 8.6) und ihr Menü auf höherer Ebene anzeigen. Das Hauptgerät muss sicherstellen, dass eine solche Sitzung zum Erstellen verfügbar ist, wenn es das Objekt 'Menü Eingeben' sendet.

8.4.2.1 Anforderung Anwendungsinformation (application_info_enq)

Tabelle 20 – Codierung des Objektes Anforderung Anwendungsinformation

Syntax	Anzahl der Bits	Mnemonik
application_info_enq () {		
application_info_enq_tag	**24**	**uimsbf**
length_field()=0		
}		

8.4.2.2 Anwendungsinformation (application_info)

Tabelle 21 – Codierung des Objektes Anwendungsinformation

Syntax	Anzahl der Bits	Mnemonik
application_info () {		
application_info_tag	**24**	**uimsbf**
length_field()		
application_type	**8**	**uimsbf**
application_manufacturer	**16**	**uimsbf**
manufacturer_code	**16**	**uimsbf**
menu_string_length	**8**	**uimsbf**
for (i=0; i<menu_string_length; i++) {		
text_char	**8**	**uimsbf**
}		
}		

application_type (Anwendungs-Typ)

Anwendungstyp	Anwendungstyp-Wert
Bedingter Zugriff	01
Elektronischer Programmführer	02
Reserviert	Weitere Werte

application_manufacturer (Anwendungs-Hersteller)

Werte für dieses Feld werden von den in [5] definierten CA-System-Kennungswerten abgeleitet.

manufacturer_code (Hersteller-Code)

Der Inhalt dieses Feldes wird vom Hersteller nach seinen Wünschen definiert.

menu_string_length (Menü-Zeichenketten-Länge)

Alle Anwendungen haben einen Benutzer-Menübaum, von dem dies die Einsprungstelle auf höherer Ebene ist und der als Teilbaum im eigenen Menübaum des Hauptgerätes vorhanden ist. Auf ihn folgt eine Zeichenfolge, die den Titel des Menüeintrags angibt. Die Struktur des eigenen Menübaumes des Hauptgerätes kann frei gewählt werden, es kann jedoch das Feld application_type benutzt werden, um Menüeinträge ähnlicher Anwendungen zu gruppieren. Das 'Menü' kann praktisch eine einfache Anzeige ohne Benutzerinteraktion oder ein komplexer Satz von Menübildschirmen sein, die anspruchsvolle Benutzerinteraktion ermöglichen.

text_char (Textzeichen)

Textinformation wird unter Verwendung der Zeichensätze und Verfahren codiert, die in [4] beschrieben sind.

8.4.2.3 Menü Eingeben (enter_menu)

Tabelle 22 – Codierung des Objekts Menü Eingeben

Syntax	Anzahl der Bits	Mnemonik
enter_menu () {		
enter_menu_tag	**24**	**uimsbf**
length_field()=0		
}		

8.4.3 Unterstützung der Zugriffsbeschränkung

Dieses Hilfsmittel stellt eine Reihe von Objekten speziell für die Unterstützung von Anwendungen für die Zugriffsbeschränkung zur Verfügung. Wie der Hilfsmittelmanager wird es nur vom Hauptgerät geliefert und hat keine Sitzungsbegrenzung. Alle CA-Anwendungen erzeugen eine Sitzung mit diesem Hilfsmittel, sobald sie ihre Anwendungsinformationsphase der Initialisierung abgeschlossen haben. Das Hauptgerät sendet ein Objekt 'Anfrage CA-Information' zu der Anwendung, die durch Rücksenden eines Objektes 'CA-Information' mit der dazugehörenden Information antwortet. Die Sitzung bleibt dann für periodischen Betrieb des Protokolls offen, das mit den Objekten CA-PMT und CA-PMT Reply verbunden ist.

8.4.3.1 Anforderung CA-Information (ca_info_enq)

Tabelle 23 – Codierung des Objektes Anforderung CA-Information

Syntax	Anzahl der Bits	Mnemonik
ca_info_enq () {		
ca_info_enq_tag	**24**	**uimsbf**
length_field()=0		
}		

8.4.3.2 CA-Information (ca_info)

Tabelle 24 – Codierung des Objektes CA-Information

Syntax	Anzahl der Bits	Mnemonik
ca_info () {		
ca_info_tag	**24**	**uimsbf**
length_field()		
for (i=0; i<N; i++) {		
CA_system_id	**16**	**uimsbf**
}		
}		

CA_system_id (CA-System-Kennung)

Sie listet die CA-System-Kennungen auf, die von dieser Anwendung unterstützt werden. Werte für CA-System-Kennungen werden in [5] definiert.

8.4.3.3 Auswählen von zu descrambelnden Diensten

CA PMT (CA-Programmtabellen) werden vom Hauptgerät zu einer oder mehreren angeschlossenen CA-Anwendungen gesendet, um anzuzeigen, welche Elementarströme vom Benutzer gewählt wurden und wie die entsprechenden ECMs (Berechtigungssteuerungsmeldungen) zu finden sind. Jedes Objekt CA PMT enthält Hinweise auf die gewählten Elementarströme eines gewählten Programms. Wenn vom Benutzer mehrere Programme gewählt werden, dann werden mehrere Objekte CA PMT gesendet. Das Hauptgerät darf die CA PMT an alle angeschlossenen CA-Anwendungen senden, oder vorzugsweise nur an die Anwendungen, die denselben CA_system_id-Wert unterstützen wie der Wert, der in dem CA-Deskriptor der gewählten Elementarströme (ES) angegeben wird.

Jede Anwendung antwortet dann auf Anforderung des Hauptgeräts mit der Antwort 'CA-Programmtabelle', die dem Hauptgerät ermöglicht, das Modul auszuwählen, das das Descrambeln ausführen wird.

8.4.3.4 CA-Programmtabelle (ca_pmt)

Das Objekt CA PMT ist eine Tabelle, die von dem Hauptgerät aus der Programmtabelle (PMT) in der SI-Information (siehe [1] 2.4.4.8 und 2.4.4.9) ausgewählt und zu der Anwendung gesendet wird. Diese Tabelle enthält alle Informationen über die Zugriffssteuerung, die der Anwendung erlaubt, die ECMs selbst herauszufiltern und die korrekte Zuweisungen eines ECM-Stromes zu einer gescrambelten Komponente selbst durchzuführen.

ca_pmt_cmd_id /*auf Programmebene */ **8 uimsbf**

Tabelle 25 – Codierung des Objektes CA-Programmtabelle

Syntax	Anzahl der Bits	Mnemonik
ca_pmt () {		
ca_pmt_tag	**24**	**uimsbf**
length_field()		
ca_pmt_list_management	**8**	**uimsbf**
program_number	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
reserved	**4**	**bslbf**
program_info_length	**12**	**uimsbf**
if (program_info_length !=0) {		
for (i=0; i<n; i++) {		
CA_descriptor() /* CA-Deskriptor auf Programmebene */		
}		
}		
for (i=0; i<n; i++) {		
stream_type	**8**	**uimsbf**
reserved	**3**	**bslbf**
elementary_PID /* Elementarstrom-PID */	**13**	**uimsbf**
reserved	**4**	**bslbf**
ES_info_length	**12**	**uimsbf**
if (ES_info_length != 0) {		
ca_pmt_cmd_id /*auf ES-Ebene */	**8**	**uimsbf**
for (i=0; i<n; i++) {		
CA_descriptor() /* CA-Deskriptor auf Elementarstromebene */		
}		
}		
}		

Die CA PMT enthält alle CA-Descriptoren des gewählten Programms. Wenn mehrere Programme gewählt werden, sendet das Hauptgerät mehrere Objekte CA PMT an die Anwendung. Die CA PMT enthält nur CA_Descriptoren. Alle anderen Descriptoren müssen durch das Hauptgerät aus der PMT entfernt werden.

Außer für ca_pmt_list_management und ca_pmt_cmd_id, die nachstehend beschrieben werden, werden alle Felder der CA PMT in [1] in 2.4.4.8 und 2.4.4.9 beschrieben.

Der in dieser Beschreibung benutzte CA-Deskriptor ist der in [1] 2.6.16 definierte CA_descriptor().

Den CA_descriptor nach dem current_next_indicator gibt es auf Programmebene und er ist für alle elementaren Komponenten des Programms gültig. Die CA-Dekriptoren auf Elementarstromebene sind nur für Elementarströme gültig. Wenn es für einen Elementarstrom CA-Deskriptoren auf Programmebene und Elementarstromebene gibt, wird nur der CA-Deskriptor auf Elementarstromebene berücksichtigt.

Der Empfänger sendet eine neue CA PMT oder eine neue Liste von CA PMT zu den Anwendungen wenn:

– der Benutzer ein anderes Programm wählt;
– ein 'tune'-Befehl weitere Dienste wählt (siehe 8.5.1.1);
– die version_number (Versionsnummer) sich ändert;
– der current_next_indicator (aktueller Next-Indikator) sich ändert.

ca_pmt_list_management (CA-Programmtabellenlisten-Verwaltung)

Dieser Parameter wird benutzt, um anzuzeigen, ob der Benutzer ein einzelnes Programm (bestehend aus einem oder mehreren Elementarströmen) oder mehrere Programme benutzt. Die nachstehenden Werte können verwendet werden:

ca_pmt_list_management		Wert
more	Mehr	00
first	Erstes	01
last	Letztes	02
only	Ausschließlich	03
add	Hinzufügen	04
update	Aktualisieren	05
reserved	Reserviert	Weitere Werte

Auf 'erstes' gesetzt bedeutet, dass die CA PMT die erste einer neuen Liste von mehr als einem Objekt CA PMT ist. Alle vorhergehend gewählten Programme sind durch die Programme der neuen Liste zu ersetzen. 'Mehr' bedeutet, dass die CA PMT weder die erste noch die letzte der Liste ist. 'Letztes' bedeutet, dass das Objekt CA PMT das letzte der Liste ist. 'Ausschließlich' bedeutet, dass die Liste aus einer einzigen CA PMT besteht. 'Hinzufügen' bedeutet, dass diese CA PMT zu einer bestehenden Liste hinzugefügt werden muss, d. h. vom Benutzer muss ein neues Programm gewählt werden, aber alle vorher gewählten Programme bleiben gewählt. Wenn für ein schon vorhandenes Programm 'hinzufügen' empfangen wird, dann ist seine Wirkung mit 'aktualisieren' identisch. 'Aktualisieren' bedeutet, dass die CA PMT eines schon in der Liste befindlichen Programms wieder gesendet wird, weil die Versionsnummer oder der aktuelle Next-Indicator sich geändert haben. Die Anwendung ist dafür zuständig zu überprüfen, ob die Änderung der Versionsnummer aus einer Änderung der CA-Operationen resultiert oder nicht. Da die Liste Management-Befehle nur auf Programmebene wirkt, muss jede Änderung auf der Elementarstromebene in einem vorhandenen Programm durch einen Befehl 'aktualisieren' mit der zurückgeschickten vollständigen Elementarstromliste signalisiert werden.

ca_pmt_cmd_id (CA-Programmtabelle-Befehlskennung)
Dieser Parameter zeigt an, welche Antwort von der Anwendung an ein Objekt CA PMT gefordert wird. Sie kann die nachstehenden Werte annehmen:

ca_pmt_cmd_id		ca_pmt_cmd_id-Werte
ok_descrambling	Descrambeln in Ordnung	01
ok_mmi	MMI in Ordnung	02
query	Rückfrage	03
not selected	Nicht gewählt	04
RFU	Reserviert für zukünftige Verwendung	Weitere Werte

Auf 'ok_descrambling' gesetzt bedeutet, dass das Hauptgerät auf die CA PMT keine Antwort erwartet und die Anwendung beginnen kann, das Programm zu descrambeln oder unmittelbar einen MMI(Mensch-Maschine-Schnittstelle)-Dialog zu beginnen. Auf 'ok_mmi' gesetzt bedeutet, dass die Anwendung einen MMI-Dialog beginnen kann, aber das Descrambeln nicht beginnen darf, ehe ein neues Objekt CA PMT mit auf 'ok_descrambling' gesetzter ca_pmt_cmd_id empfangen wurde. In diesem Fall muss das Hauptgerät sicherstellen, dass durch die CA-Anwendung eine MMI-Sitzung eröffnet werden kann. Auf 'Rückfrage' gesetzt bedeutet, dass das Hauptgerät erwartet, eine CA PMT-Antwort zu empfangen. Dann ist es der Anwendung nicht erlaubt, das Descrambeln oder einen MMI-Dialog zu beginnen, bevor ein neues Objekt CA PMT mit auf 'ok_descrambling' oder 'ok_mmi' gesetzter ca_pmt_cmd_id empfangen wurde. Auf 'nicht gewählt' gesetzt zeigt sie der CA-Anwendung an, dass das Hauptgerät nicht länger anfordert, dass die CA-Anwendung versucht, den Dienst zu descrambeln. Die CA-Anwendung muss jeden MMI-Dialog schließen, den sie eröffnet hat.

8.4.3.5 CA PMT Antwort (ca_pmt_reply)
Dieses Objekt wird immer nach Empfang eines Objektes CA PMT mit auf 'query' gesetzter ca_pmt_cmd_id von der Anwendung an das Hauptgerät gesendet. Es kann auch nach Empfang eines Objektes CA PMT mit auf 'ok_mmi' gesetzter ca_pmt_cmd_id gesendet werden, um dem Hauptgerät das Ergebnis des MMI-Dialoges anzuzeigen ('Descrambeln möglich', wenn der Benutzer gekauft hat, 'Descrambeln nicht möglich (weil keine Berechtigung)', wenn der Benutzer nicht gekauft hat).

Tabelle 26 – Codierung des Objektes CA PMT Antwort

Syntax	Anzahl der Bits	Mnemonik
ca_pmt_reply() {		
ca_pmt_reply_tag	**24**	**uimsbf**
length_field()		
program_number	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
CA_enable_flag	**1**	**bslbf**
if (CA_enable_flag == 1)		
CA_enable /* auf Programmebene */	**7**	**uimsbf**
else if (CA_enable_flag == 0)		
reserved	**7**	**bslbf**
for (i=0; i<n; i++) {		
reserved	**3**	**bslbf**
elementary_PID /* Elementarstrom-PID */	**13**	**uimsbf**
CA_enable_flag	**1**	**bslbf**
if (CA_enable_flag == 1)		
CA_enable /* wahlfr. CA_enable auf ES-Ebene*/	**7**	**uimsbf**
else if (CA_enable_flag == 0)		
reserved	**7**	**bslbf**
}		
}		

Die Syntax enthält ein mögliches ca_enable auf Programmebene und für jeden Elementarstrom ein mögliches ca_enable auf Elementarstromebene.

– Wenn beide vorhanden sind, gilt für diesen Elementarstrom nur ein ca_enable auf ES-Ebene.

– Wenn keines vorhanden ist, interpretiert das Hauptgerät das ca_pmt_reply-Objekt nicht.

Das CA_enable(CA-freigeben)-Feld zeigt an, ob die Anwendung in der Lage ist, die Descrambeloperation durchzuführen. CA_enable-Werte werden wie folgt codiert:

CA_enable	Wert
Descrambeln möglich	01
Descrambeln unter Bedingungen möglich (Kaufdialog)	02
Descrambeln unter Bedingungen möglich (technischer Dialog)	03
Descrambeln nicht möglich (weil keine Berechtigung)	71
Descrambeln nicht möglich (aus technischen Gründen)	73
Reserviert für zukünftige Verwendung	Weitere Werte 00 bis 7F

Der Wert 'Descrambeln möglich' bedeutet, dass die Anwendung ohne besondere Bedingungen descrambeln kann (z. B. der Benutzer hat ein Abonnement) oder der Benutzer hat den Kauf des Elementarstromes genehmigt.

Der Wert 'Descrambeln unter Bedingungen möglich (Kaufdialog)' bedeutet, dass die Anwendung in einen Verkaufsdialog mit dem Benutzer eintreten muss, ehe sie descrambeln kann (pay per view-Programme).

Der Wert 'Descrambeln unter Bedingungen möglich (technischer Dialog)' bedeutet, dass die Anwendung in einen technischen Dialog mit dem Benutzer eintreten muss, ehe sie descrambeln kann (fordert z. B. den Benutzer auf, weniger Elementarströme zu benutzen, weil die Descrambelkapazitäten begrenzt sind).

Der Wert 'Descrambeln nicht möglich (weil keine Berechtigung)' bedeutet, dass der Benutzer für das Programm keine Berechtigung hat oder der Benutzer das Programm nicht kaufen will.

Der Wert 'Descrambeln nicht möglich (aus technischen Gründen)' bedeutet, dass die Anwendung den Elementarstrom aus technischen Gründen nicht descrambeln kann (z. B. alle Descrambelkapazitäten sind schon in Benutzung).

Das Protokoll lässt Dienste zu, die für das Descrambeln entweder auf Programmebene oder auf Elementarstromebene auszuwählen sind. Wo das Hauptgerät das Descrambeln von unterschiedlichen Elementarströmen durch verschiedene Module nicht unterstützen kann, muss es als CA_enable-Wert für das Programm den niedrigsten der CA_enable-Werte nehmen, die für jeden Elementarstrom des Programms geantwortet werden. Dies stellt sicher, dass jeder Elementarstrom descrambelt wird, wenn er descrambelt werden kann.

8.5 Hilfsmittel Hauptgerätesteuerung und Information

8.5.1 DVB-Hauptgerätesteuerung

Das Hilfsmittel 'DVB-Hauptgerätesteuerung' ermöglicht einer Anwendung eine begrenzte Steuerung des Betriebs des Hauptgerätes. In dieser Version gibt es zwei Hauptfunktionen. Mit der ersten Funktion wird das Hauptgerät auf einen anderen Dienst abgestimmt. Dies kann eine Frequenzänderung sein und sogar das Einstellen auf einen anderen Satelliten. Kontext mit dem vorhergehenden Zustand kann nicht beibehalten werden. Mit der zweiten Funktion wird ein Dienst zeitweise durch einen anderen ersetzt. Dies kann von kurzer Dauer sein, wie beim Ersetzen von Werbung, oder von längerer Dauer, wie bei einem ganzen Programm. Die erste Funktion wird durch ein Objekt 'Abstimmen' unterstützt. Die zweite wird durch 'Ersetzen' und 'Ersetzen aufheben' unterstützt. Für das Hauptgerät gibt es auch ein Objekt 'Freigabe anfragen', um damit eine Anwendung aufzufordern, die Sitzung zu schließen. Nur das Hauptgerät stellt das Hilfsmittel 'DVB-Hauptgerätesteuerung' zur Verfügung, und es kann zu einem Zeitpunkt nur eine Sitzung unterstützen.

8.5.1.1 Abstimmen (tune)

Erlaubt der Anwendung, das Hauptgerät auf einen anderen Dienst abzustimmen. Die Parameter dieses Objektes, Sendernetzkennung, Ursprungs-Sendernetzkennung, Transportstromkennung und Dienstkennung werden in [4] definiert. Wenn das Hauptgerät auf den neuen Dienst abgestimmt ist, muss es unter Benutzung des Objektes CA_PMT (8.4.4) in den Standard-CA-Unterstützungsdialog eintreten, um den neuen Dienst descrambeln zu können. Das Senden dieses Objektes kann bewirken, dass das Hauptgerät seinen augenblicklichen Zustand verliert. Dann wird das Hauptgerät nicht auf den vorhergehenden Dienst zurück abgestimmt, wenn die DVB-Hauptgerätesteuerungs-Sitzung geschlossen ist.

Tabelle 27 – Codierung des Objektes Abstimmen

Syntax	Anzahl der Bits	Mnemonik
tune () {		
tune_tag	**24**	**uimsbf**
length_field()		
network_id	**16**	**uimsbf**
original_network_id	**16**	**uimsbf**
transport_stream_id	**16**	**uimsbf**
service_id	**16**	**uimsbf**
}		

8.5.1.2 Ersetzen (replace)

Replace und clear replace (Ersetzen aufheben) ermöglichen, eine Servicekomponente zeitweise durch eine andere derselben Multiplexart zu ersetzen. Die Anwendung weist einen replacement_ref-Wert zu, der dazu dient, ein Objekt clear replace an ein oder mehrere vorhergehende Objekte replace anzupassen. Die Anwendung sendet dann ein oder mehrere Objekte replace, um anzufordern, dass eine Komponente in der replaced_PID – Video, Audio, Teletext oder Untertitel – durch die Komponente ersetzt wird, die in der replacement_PID übertragen wurden. Das Ersetzen erfolgt unmittelbar. Mehrere Objekte replace können denselben Wert der replacement_ref benutzen. In diesem Fall werden sie alle aufgehoben, wenn das entsprechende Objekt clear replace gesendet wird. Das Hauptgerät muss den Kontext zu den vorhergehenden Diensten festhalten, so dass es sie wieder einsetzen kann, wenn clear replace gesendet wird. Der vorhergehende Kontext wird, wenn möglich, auch wieder hergestellt, wenn die Sitzung mit dem Hilfsmittel 'DVB-Hauptgerätesteuerung' geschlossen wird.

Tabelle 28 – Codierung des Objektes Ersetzen

Syntax	Anzahl der Bits	Mnemonik
replace () {		
replace_tag	**24**	**uimsbf**
length_field()		
replacement_ref	**8**	**uimsbf**
reserved	**3**	**bslbf**
replaced_PID	**13**	**uimsbf**
reserved	**3**	**bslbf**
replacement_PID	**13**	**uimsbf**
}		

8.5.1.3 Ersetzen Aufheben (clear replace)

Tabelle 29 – Codierung des Objektes Ersetzen Aufheben

Syntax	Anzahl der Bits	Mnemonik
clear_replace () {		
clear_replace tag	**24**	**uimsbf**
length_field()		
replacement_ref	**8**	**uimsbf**
}		

8.5.1.4 Freigabe Anfragen (ask release)

Solange eine Anwendung eine Sitzung mit dem Hilfsmittel 'DVB-Hauptgerätesteuerung' durchführt, hat sie die Kontrolle über Teile des Verhaltens des Hauptgerätes. Es kann für das Hauptgerät erforderlich sein, die Steuerung zurückzubekommen. Dazu sendet es das Objekt 'ask release' an die Anwendung. Der Anwendung wird eine kurze überwachte Zeitspanne gegeben, um gegebenenfalls 'clear replace'-Objekte zu senden und die Sitzung zu schließen. Wenn die Sitzung am Ende der Zeitüberwachung nicht geschlossen ist, wird das Hauptgerät die Sitzung trotzdem schließen und, wenn möglich, den vorhergehenden Zustand des Hauptgerätes wieder herstellen.

Tabelle 30 – Codierung des Objektes Freigabe Anfragen

Syntax	Anzahl der Bits	Mnemonik
ask_release () {		
ask_release_tag	**24**	**uimsbf**
length_field() = 0		
}		

8.5.2 Datum und Zeit

Das Hilfsmittel Datum und Zeit wird vom Hauptgerät bereitgestellt und kann unbegrenzte Sitzungen unterstützen. Eine Anwendung erstellt eine Sitzung mit dem Hilfsmittel und fragt dann mit einem Objekt 'date_time_enq' die augenblickliche Zeit an. Wenn 'response_interval' gleich Null ist, dann erfolgt die Antwort als einzelnes Objekt 'date_time' sofort. Wenn 'response_interval' ungleich Null ist, dann erfolgt die Antwort als ein Objekt 'date_time' sofort, gefolgt von weiteren 'date_time'-Objekten alle 'response_interval'-Sekunden. In einem DVB-konformen Hauptgerät wird die bereitgestellte Zeit von der Zeit- und Datumtabelle (TDT) in der SI-Information abgeleitet (siehe [4]).

8.5.2.1 Anfrage Datum-Zeit (date_time_enq)

Tabelle 31 – Codierung des Objektes Anfrage Datum-Zeit

Syntax	Anzahl der Bits	Mnemonik
date_time_enq () {		
date_time_enq_tag	**24**	**uimsbf**
length_field()		
response_interval	**8**	**uimsbf**
}		

8.5.2.2 Datum-Zeit (date_time)

Tabelle 32 – Codierung des Objektes Datum-Zeit

Syntax	Anzahl der Bits	Mnemonik
date_time () {		
date_time_tag	**24**	**uimsbf**
length_field()= 5 or 7		
UTC_time	**40**	**bslbf**
local_offset /* freigestellt */	**16**	**tcimsbf**
}		

UTC_time (UTC-Zeit)
UTC_time überträgt das UTC-Datum und die UTC-Zeit der nächsten Sekunde, codiert als modifizierter julianischer Tag und Stunden, Minuten und Sekunden, wie in [4] beschrieben.

local_offset (Zeitversatz)
Dies ist ein wahlfreies Feld. Wenn vorhanden, codiert es den augenblicklichen Versatz zwischen UTC und Ortszeit als eine mit Vorzeichen versehene Zahl von Minuten. Ortszeit = UTC_time + local_offset. Das Hauptgerät muss das local_offset-Feld bereitstellen, wenn es zuverlässige Kenntnis seines Wertes hat. Andernfalls muss es ausgelassen werden.

8.6 Hilfsmittel Mensch-Maschine-Schnittstelle (MMI)

8.6.1 Einleitung

Es wird eine Hilfsmittelklasse zur Verfügung gestellt. Sie unterstützt Dialoge von Anzeige und Tastatur mit dem Benutzer. Es werden zwei Dialogebenen definiert. Im MMI-Mode auf unterer Ebene gibt die Anwendung ausführliche Steuerungsmöglichkeiten bezüglich dieses Dialoges, einschließlich Empfangs-Fernbedienungs-Tastencodes direkt vom Benutzer mit der Möglichkeit, das Layout und andere Attribute der Bildschirmanzeige im Einzelnen zu steuern. Im MMI-Mode auf höherer Ebene gibt die Anwendung einen Satz von Objekten mit Semantik auf höherer Ebene, einschließlich Menüs und Listen aus. In diesem Fall bestimmt das Hauptgerät das Aussehen und die Art und Weise der Anzeige. Es ist nicht möglich, in derselben Sitzung MMI-Modes zu mischen. Ob das Hauptgerät gleichzeitig mehr als eine MMI-Sitzung unterstützen kann, hängt von der Konzeption des Hauptgerätes ab.

Text, der von Anzeigeobjekten im Mode auf höherer Ebene ausgegeben wird, wird entsprechend [4] codiert. Text, der von Anzeigeobjekten im Mode auf unterer Ebene ausgegeben wird, wird entsprechend [9] codiert.

8.6.2 In beiden Moden benutzte Objekte

8.6.2.1 MMI Schließen (close MMI)

Dieses Objekt kann vom Hauptgerät oder Modul gesendet werden, um dem Benutzer oder einer Anwendung zu ermöglichen, jeden Dialog im MMI-Mode auf höherer oder unterer Ebene zu verlassen.

Tabelle 33 – Codierung des Objektes MMI Schließen

Syntax	Anzahl der Bits	Mnemonik
close_mmi() {		
close_mmi_tag	**24**	**uimsbf**
length_field()		
close_mmi_cmd_id	**8**	**uimsbf**
if (close_mmi_smd_id == delay) {		
delay	**8**	**uimsbf**
}		
}		

close_mmi_cmd_id (MMI Schließen Befehlskennung)

close_mmi_cmd_id		Wert
immediate	Sofort	00
delay	Verzögert	01
reserved	Reserviert	Andere Werte

Zeigt an, ob die Anzeige sofort zu dem vorhergehenden Dienst zurückgeschaltet werden sollte oder ob verzögert werden sollte, um zu ermöglichen, dass ein weiterer MMI-Dialog diesen ersetzt.

delay (Verzögerung)

Die Verzögerung in Sekunden, bevor die Anzeige zu dem laufenden Dienst zurückgeschaltet werden sollte, wenn in der Zwischenzeit keine weitere MMI-Sitzung begonnen wurde.

Wenn delay von einer Anwendung gesendet wird, muss das Hauptgerät sofort die laufende MMI-Sitzung für die Anwendung schließen. Wenn close_mmi_cmd_id 'immediate' ist, dann muss das Hauptgerät auch sofort zu dem vorherigen Anzeigezustand zurückkehren. Wenn close_mmi_cmd_id 'delay' ist, dann muss das Hauptgerät den letzten Bildschirmzustand der MMI-Sitzung beibehalten, bis entweder die spezielle Verzögerung abläuft oder bis eine weitere MMI-Sitzung gestartet wird. Die Absicht ist, einen nahtlosen Wechsel des MMI-Dialoges von einer Anwendung zu einer anderen zu ermöglichen, ohne dass kurze Stücke des laufenden Dienstes dazwischen auftreten. Wenn die Anwendung die Sitzung mit dem MMI-Hilfmittel schließt anstatt dieses Objekt zu benutzen, dann muss das Hauptgerät es als close_mmi mit dem Befehl 'intermediate' interpretieren.

Wenn vom Hauptgerät gesendet, muss die Anwendung die laufende MMI-Sitzung schließen. Diese Art wird bevorzugt demgegenüber durchgeführt, dass nur das Hauptgerät die Sitzung mit dem MMI-Hilfsmittel schließen kann, um der Anwendung zu ermöglichen, auf geeignete Art abzuschalten. In diesem Fall ist die Verzögerungsfunktion unnötig, und das Hauptgerät muss immer den Befehl 'immediate' benutzen.

8.6.2.2 Anzeigesteuerung (display control)

Das Objekt Anzeigesteuerung ermöglicht einer Anwendung, beim Hauptgerät nach dessen Graphik/Anzeige-Fähigkeiten in drei Betriebsarten anzufragen:

– Bitmapgraphiken, über Video überlagert;

– Bitmapgraphiken, Video ersetzend;

– Zeichen, auf MMI-Mode höherer Ebene basierend.

Diese Kommunikationen ermöglichen der Anwendung zu entnehmen:

– das adressierbare Koordinatensystem der Anzeige (im Prinzip um 625- oder 525-Zeilen-Anzeigen zu unterscheiden);

– das Bildseitenverhältnis der Bildpunkte;

– die Anzahl der Bildpunkte/Bildpunkttiefe, die die Anwendung benutzen kann;

– verfügbare Zeichensätze.

Das Objekt Anzeigesteuerung führt zwei Funktionen aus. Eine Funktion fordert Informationen über die Eigenschaften beim Anzeigen von Information. Die zweite Funktion dient zum Einstellen der Betriebsart der Anzeige beim Anzeigen von Information.

Tabelle 34 – Codierung des Objektes Anzeigesteuerung

Syntax	**Anzahl der Bits**	**Mnemonik**
display_control() {		
display_control_tag	**24**	**uimsbf**
length_field()		
display_control_cmd	**8**	**uimsbf**
if (display_control_cmd == set_MMI_mode) {		
MMI_mode	**8**	**uimsbf**
}		
}		

Die Anzeige-Steuerungsbefehle (display_control_cmd) werden nachstehend definiert:

display_control_cmd	Wert	Beschreibung des Befehls
set_mmi_mode	01	Fordert an, dass das Hauptgerät den durch das MMI_mode-Byte angezeigten Mode eingibt
get_display_character_table_list	02	Fordert an, dass das Hauptgerät mit einer Liste der Zeichencodetabellen antwortet, die es während der Anzeigevorgänge unterstützen kann
get_input_character_table_list	03	Fordert an, dass das Hauptgerät mit einer Liste der Zeichencodetabellen antwortet, die es während der Eingangsvorgänge unterstützen kann
get_overlay_graphics_characteristics	04	Fordert das Profil der Anzeige an, wenn für die Anzeige Graphik über Video überlagert benutzt wird
get_full_screen_graphics_characteristics	05	Fordert das Profil der Anzeige an, wenn für die Anzeige Graphik als Ersatz für Video benutzt wird
reserved	Weitere Werte	

mmi_mode	Wert	Beschreibung
Höhere Ebene	01	Fordert an, dass eine MMI-Sitzung auf höherer Ebene eröffnet wird. Wenn sie auf der Haupt-Videoanzeige implementiert wird, kann dies jedes Bild, das augenblicklich angezeigt wird, teilweise oder vollständig verdecken
Überlagerte Graphik auf unterer Ebene	02	Fordert an, dass eine graphische MMI-Sitzung auf unterer Ebene eröffnet wird, die (eine gegebenenfalls aktive) Haupt-Videoanzeige überlagert
Ganze Bildschirm-Graphik auf unterer Ebene	03	Fordert an, dass eine graphische MMI-Sitzung auf unterer Ebene eröffnet wird, die die Haupt-Videoanzeige ersetzt (oder von der Haupt-Videoanzeige unabhängig ist)
Reserviert	Weitere Werte	

8.6.2.3 Anzeigeantwort (display_reply)

Das Objekt Anzeigeantwort ist die Antwort vom Anzeigesystem des Hauptgerätes auf die Objekte Anzeigesteuerung, die von der Anwendung gesendet werden.

Tabelle 35 – Codierung des Objektes Anzeigeantwort

Syntax	Anzahl der Bits	Mnemonik
display_reply() {		
display_reply_tag	**24**	**uimsbf**
length_field()		
display_reply_id	**8**	**uimsbf**
if (display_reply_id == list_graphic_overlay_characteristics ||		
display_reply_id == list_full_screen_graphic_characteristics) {		
display_horizontal_size	**16**	**uimsbf**
display_vertical_size	**16**	**uimsbf**
aspect_ratio_information	**4**	**uimsbf**
graphics_relation_to_video	**3**	**bslbf**
multiple_depths	**1**	**bslbf**
display_bytes	**12**	**uimsbf**
composition_buffer_bytes	**8**	**uimsbf**
object_cache_bytes	**8**	**uimsbf**
number_pixel_depths	**4**	**uimsbf**
for (i=0;i<n;i++) {		
display_depth	**3**	**uimsbf**
pixels_per_byte	**3**	**uimsbf**
reserved	**2**	**bslbf**
region_overhead	**8**	**uimsbf**
}		
if (display_reply_id == list_display_character_tables ||		
display_reply_id == list_input_character_tables) {		
for (i=0;i<n;i++) {		
character_table_byte	**8**	**uimsbf**
}		
}		
if (display_reply_id == mmi_mode_ack) {		
/* Bestätigung des gewählten MMI-Mode */		
mmi_mode	**8**	**uimsbf**
}		
}		

display_reply_id (Anzeigeantwortkennung)
Wird wie folgt codiert:

display_reply_id	Anzeigeantwortkennung	id-Werte
mmi_mode_ack	MMI-Mode-Bestätigung	01
list_display_character_tables	Liste der Anzeige-Zeichentabellen	02
list_input_character_tables	Liste der Eingangs-Zeichentabellen	03
list_graphic_overlay_characteristics	Liste der Graphik-Überlagerungsmerkmale	04
list_full_screen_graphic_characteristics	Liste der Gesamtbildschirm-Graphikmerkmale	05
unknown_display_control_cmd	Unbekannter Anzeige-Steuerungsbefehl	F0
unknown_mmi_mode	Unbekannter MMI-Mode	F1
unknown_character_table	Unbekannte Zeichentabelle	F2
reserved	Reserviert	Weitere Werte

display_horizontal_size (horizontale Anzeigegröße)

display_vertical_size (vertikale Anzeigegröße)

Diese (ganzzahligen) 16-Bit-Zahlen beschreiben den maximalen adressierbaren Koordinatenbereich der Bitmapgraphikanzeige. Alle Positionen von (0,0) bis zu (horizontal Anzeigegröße – 1, vertikale Anzeigegröße – 1) können adressiert werden. Alle Decoder sollten mindestens 720 Pixelspalten unterstützen. Die Anzahl der zur Verfügung stehenden Abtastzeilen sollte mindestens der vertikalen Auflösung des im Augenblick decodierten Videoformates entsprechen. So sollten beim 625/50-Videoformat mindestens 576 Zeilen zur Verfügung stehen bzw. beim 525/60-Videoformat 480 Zeilen.

aspect_ratio_information (Information über das Bildseitenverhältnis)

Dieses 4-Bit-Feld wird auf die gleiche Art codiert wie das aspect_ratio_information-Feld von ISO/IEC 1318-2. Es ermöglicht, aus der horizontalen Anzeigegröße und der vertikalen Anzeigegröße das Pixel-Bildseitenverhältnis zu bestimmen.

graphics_relation_to_video (Verhältnis Graphik zu Video)

Dieses 3-Bit-Feld zeigt den Bereich an, in dem die Videoebene der Graphikebene entspricht. Dies zeigt an, ob Graphiken bezüglich des Videobildes vorhersehbar positioniert werden können. Der Rundfunkbetreiber sollte wissen, dass jede Fernsehsendung mit reduzierter Auflösung durch Nachbearbeiten im Videodecoder in einer von der Implementierung abhängigen Art skaliert werden kann. Im Grenzfall könnte die Graphik auf einer anderen Anzeigeeinrichtung als der Bild-Anzeigeeinrichtung dargestellt werden.

graphics_relation_to_video	Wert
Keine Beziehung zwischen Graphik und Video	000
Reserviert	001 bis 110
Die Graphik-Koordinatenebene stimmt exakt mit der Video-Koordinatenebene überein	111

multiple_depths (mehrere Tiefen)

Auf '1' gesetzt, zeigt dieses 1-Bit-Feld an, ob die Anzeige die Mischung von unterschiedlichen Pixeltiefen ist (d. h. jeder Anzeigebereich könnte mit einer anderen Pixeltiefe angegeben werden). Auf '0' gesetzt, zeigt dieses Feld an, dass das Anzeigesystem beim Mischen unterschiedlicher Pixeltiefen (nicht angegebene) Beschränkungen hat. In diesem Fall sollte für alle Bereiche nur eine einzige Pixeltiefe benutzt werden.

display_bytes (Anzeige-Bytes)

Diese (ganzzahlige) 12-Bit-Zahl gibt, wenn sie mit 256 multipliziert wird, die Anzahl der für den Anzeigespeicher verfügbaren Bytes an.

composition_buffer_bytes (Seitenaufbau-Puffer-Bytes)

Diese (ganzzahlige) 8-Bit-Zahl gibt, wenn sie mit 256 multipliziert wird, die Anzahl der für den Seitenaufbau-Puffer verfügbaren Bytes an.

objekt_cache_bytes (Objekt-Cache-Bytes)

Diese (ganzzahlige) 8-Bit-Zahl gibt, wenn sie mit 4096 multipliziert wird, die Anzahl der Bytes für den Objekt-Cache-Puffer an. Die Anforderung an den Speicher für jedes Objekt entspricht der für das DVB_Subtitling_Segment, das das Objekt ausgibt.

number_pixel_depths (Anzahl der Pixeltiefen)

Diese (ganzzahlige) 4-Bit-Zahl zeigt die Anzahl der verschiedenen Pixeltiefen an, die die Anzeige zur Verfügung stellen kann.

display_depth (Anzeigetiefe)

Dieses 3-Bit-Feld zeigt die Display-Pixeltiefe an. Sie ist auf die gleiche Art codiert wie region_level_of_compatibility in [9].

pixels_per_ byte (Pixel je Byte)

Diese (ganzzahlige) 3-Bit-Zahl gibt die Anzahl der Pixels an, die bei dieser Anzeigetiefe in jedes Byte gepackt werden können. Der Wert 0 ist ein Spezialfall, der eine 8 Bit tiefe Anzeige impliziert.

region_overhead (Bereichsaufwand)

Diese (ganzzahlige) 8-Bit-Zahl gibt, wenn mit 16 multipliziert, die Reduktion in der Anzahl der Pixel an, die angezeigt werden können, wenn ein zusätzlicher Bereich mit dieser Pixeltiefe eingeführt wird.

Der Decoder sollte in der Lage sein, die Anzeige zu implementieren, die durch die Anwendung angefordert wird, wenn die folgenden Bedingungen gelten:

Anzeigebytes ≥ benutzte Bytes

Wobei benutzte Bytes durch den nachstehend gezeigten Algorithmus berechnet werden:

```
bytes_used = 0;
for (depth=0; depth < number_pixel_depth; depth++) {
   for (region=0; region < number of regions with pixel depth == depth;
        region++) {
      bytes_used += (region_width(region, depth) *
         region_height(region, depth) / pixels_per_byte (depth);
   bytes_used += region_overhead(depth);
   }
}
```

character_table_byte (Zeichentabellenbyte)

In [4] wird die Auswahl der Zeichentabellen für Textfelder (wenn von der Vorgabe abweichend) durch das Anfangsbyte (oder Bytes) des Textfeldes angezeigt. Die Zeichentabellenbytes sind eine nichtgeordnete Liste der durch [4] definierten Zeichentabellen-Auswahlbytes.

Alle Hauptgeräte müssen Eingang und Ausgang unterstützen, die das vorgegebene Lateinische (Tabelle 0)-Alphabet benutzen wie es in [4] definiert wird.

8.6.3 MMI-Tastatur-Objekte auf unterer Ebene

Die nachstehenden Objekte werden ausschließlich im MMI-Mode auf unterer Ebene benutzt.

8.6.3.1 Tastenfeldsteuerung (keypad_control)

Die Anwendung geht von der Voraussetzung aus, dass das Hauptgerät ein Tastenfeld hat und der Benutzer diese Tastatur im Mensch-Maschine-Dialog benutzt. Das Tastenfeld kann virtuell sein: der Benutzer kann Befehle eingeben oder eine andere Art wählen (z. B. Sprache). Das Hauptgerät übersetzt diese Benutzeraktionen in virtuelle Tastenbetätigungen für die Anwendung.

Tabelle 36 – Codierung des Objektes Tastenfeldsteuerung

Syntax	Anzahl der Bits	Mnemonik
keypad_control() {		
keypad_control_tag	**24**	**uimsbf**
length_field()		
keypad_control_cmd	**8**	**uimsbf**
if (keypad_control_cmd == intercept_selected_keypress) {		
for (i=0;i<keypad_control_length – 1;i++) {		
/* Liste der akzeptierten Tastenbetätigungen */		
key_code	**8**	**uimsbf**
}		
}		
if (keypad_control_cmd == ignore_selected_keypress){		
for (i=0;i<keypad_control_length – 1;i++){		
/* Liste der ignorierten Tastenbetätigungen */		
key_code	**8**	**uimsbf**
}		
}		
if (keypad_control_cmd == reject_keypress) {		
key_code	**8**	**uimsbf**
}		
}		

Das Objekt 'keypad_control' (Tastenfeldsteuerung) und sein nachstehend beschriebenes, damit verbundenes Objekt 'keypress' (Tastenbetätigung) ermöglichen der Anwendung, virtuelle Tastenbetätigungen abzufangen. Das Objekt Tastenfeldsteuerung erlaubt der Anwendung, Tastenbetätigungen an die Anwendung zu richten. Sie werden dann über das Objekt Tastenbetätigung gesendet, wenn sie ausgeführt werden. Die Anwendung kann auch akzeptierte Tastenbetätigungen anhalten und Tastenbetätigungen zurückweisen, die sie nicht versteht oder an denen sie nicht interessiert ist. Der Satz von Tastenbetätigungen, die abgefangen werden können, ist in 8.6.3.3 definiert.

keypad_control_cmd	Tastenfeldsteuerungsbefehl	cmd-Wert
intercept_all_keypresses	Alle Tastendrücke abfangen	01
ignore_all_keypresses	Alle Tastendrücke ignorieren	02
intercept_selected_keypress	Ausgewählten Tastendruck abfangen	03
ignore_selected_keypress	Ausgewählten Tastendruck ignorieren	04
reject_keypress	Tastendruck zurückweisen	05
reserved	Reserviert	Weitere Werte

8.6.3.2 Tastenbetätigung (keypress)

Tabelle 37 – Codierung des Objektes Tastenbetätigung

Syntax	**Anzahl der Bits**	**Mnemonik**
keypress() {		
keypress_tag	**24**	**uimsbf**
length_field()= 1		
key_code	**8**	**uimsbf**
}		

8.6.3.3 Tabelle der Tastencodes

Diese Tastencodes werden an der Schnittstelle Hauptgerät/Anwendung benutzt. Die entsprechenden Tasten sind nicht notwendigerweise auf dem Tastenfeld vorhanden, das Hauptgerät muss jedoch die entsprechende Benutzeraktion in einen Tastencode übersetzen. Es ist obligatorisch, alle Tastencodes zu unterstützen.

Die Tastencodewerte (hexadezimal) sind in der folgenden Tabelle aufgeführt.

Tastencode	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
Bedeutung	0	1	2	3	4	5	6	7	8	9	Menü	ESC	⇒	⇐	⇑	⇓	BS	RC

Andere Werte (von 0x12 bis 0xFF) sind reserviert.

8.6.4 MMI-Anzeige-Objekte auf unterer Ebene

Die MMI-Anzeigen auf unterer Ebene beruhen auf der Anwendung des DVB-Untertitelungs-Mechanismus.

8.6.4.1 Ausgabe von DVB-Untertiteldaten

DVB-Untertitelungssegmente sind, wie nachstehend beschrieben wird, in APDUs der DVB-konformen Einheitliche Schnittstelle zusammengefasst. Die Folge der DVB-Untertitelsegmente sollte den in [9] definierten entsprechen. Nachdem jeder vollständige Anzeigensatz von der Anwendung zum Hauptgerät übertragen wurde, sollte die Anwendung eine Szenenendmarken-APDU übertragen. Diese APDU grenzt die Anzeigesätze ab und versieht die Anwendung mit der zeitlichen Steuerung des Decodierungsprozesses.

8.6.4.2 Segmentzusammenfassung

DVB-Untertitelungsegmente werden direkt in APDUs der DVB-kompatiblen Einheitliche Schnittstelle zusammengefasst. Das 'Mehr/Letzter'-Format wird benutzt, um lange, über mehrere APDUs zu fragmentierende Segmente zu ermöglichen.

Tabelle 38 – Codierung des Objektes Untertitelsegment (subtitle_segment)

Syntax	Anzahl der Bits	Mnemonik
subtitle_segment() {		
subtitle_segment_tag	**24**	**uimsbf**
length_field()		
DVB_Subtitling_segment()		
}		

8.6.4.3 Anzeigesystem-Nachrichten

In den Anzeigesystem-Nachrichten wird vor Situationen gewarnt, die Aufmerksamkeit erfordern.

Tabelle 39 – Codierung des Objektes Nachricht anzeigen

Syntax	Anzahl der Bits	Mnemonik
display_message() {		
display_message_tag	**24**	**uimsbf**
length_field()		
display_message_id	**8**	**uimsbf**
}		

display_message_id	Wert	verwendet als
Anzeige in Ordnung	00	Kann wahlweise vom Hauptgerät als eine positive Bestätigung jedes MMI-Objektes auf unterer oder höherer Ebene gesendet werden
Anzeigefehler	01	Wenn durch das Anzeigesystem ein Fehler festgestellt wurde
Anzeige hat keinen Speicher mehr	02	Fehleranzeige, dass das Hauptgerät seinen Seitenaufbau-Puffer, Pixel-Puffer oder Objekt-Cache-Speicher verbraucht hat
DVB-Untertitel-Syntaxfehler	03	Fehleranzeige, dass das Hauptgerät die DVB-Untertitelungs-segmente nicht interpretieren kann
Hinweis auf undefinierten Bereich	04	Fehleranzeige, dass das Hauptgerät einen Hinweis auf eine region_id gefunden hat, die noch nicht eingeführt wurde
Hinweis auf undefinierte CLUT	05	Fehleranzeige, dass das Hauptgerät einen Hinweis auf eine CLUT_id gefunden hat, die noch nicht eingeführt wurde
Hinweis auf undefiniertes Objekt	06	Fehleranzeige, dass das Hauptgerät einen Hinweis auf eine object_id gefunden hat, die noch nicht eingeführt wurde
Mit Bereich nicht kompatibles Objekt	07	Fehleranzeige, dass die Pixeltiefe oder -größe eines Objektes mit dem Bereich nicht kompatibel ist, in dem es vorkommt
Hinweis auf undefiniertes Zeichen	08	Fehleranzeige, dass ein mit der gewählten Zeichentabelle inkompatibler Zeichencode gefunden wurde
Eigenschaften der Anzeige geändert	09	Diese Nachricht zeigt an, dass einige Eigenschaften der Anzeige sich geändert haben, seitdem sie zuletzt durch die Anwendung geprüft wurden. Zum Beispiel könnte sich dies als Folge einer Benutzeraktion ergeben, wie z. B. dem Neukonfigurieren des Fernsehgerätes von 16:9- auf 4:3-Anzeige oder als Reaktion auf eine Änderung im Programmmaterial (z. B. des Anzeige-formates des Bildes durch decodierte Änderungen)
Reserviert	Andere Werte	

8.6.4.4 Zeitliche Steuerung

Wenn Untertitel gesendet werden, steuert der PTS in dem Header des PES, der die Untertitelsegmente zusammenfasst, den zeitlichen Ablauf des Decodierens und Anzeigens von aufeinander folgenden Anzeigen von Untertitelungsinformation. Wenn von der einheitlichen Schnittstelle benutzt, liefert eine Folge von APDUs zeitliche Steuerung.

Szenenendmarke (scene_end_mark)

Tabelle 40 – Codierung des Objektes Szenenendmarke

Syntax	Anzahl der Bits	Mnemonik
scene_end_mark() {		
scene_end_mark_tag	**24**	**uimsbf**
length_field()		
decoder_continue_flag	**1**	**bslbf**
scene_reveal_flag	**1**	**bslbf**
send_scene_done	**1**	**bslbf**
reserved	**1**	**bslbf**
scene_tag	**4**	**uimsbf**
}		

Die Anwendung sollte eine Szenenendmarken-APDU senden, nachdem der Satz von Untertitelsegmenten für eine Anzeige (den Anzeigensatz) zu dem Decoder gesendet wurde. Die Szenenendmarke zeigt dem Decoder das Ende des Datensatzes an und teilt ihm mit, was zu tun ist, wenn er den Satz vollständig decodiert hat.

decoder_continue_flag (Flag Decodieren fortsetzen)

Wenn dieses 1-Bit-Flag auf '1' gesetzt ist, weist es den Decoder an, das Decodieren der Untertitelungsdaten (von dieser MMI-Sitzung) fortzusetzen. Wenn es auf '0' gesetzt ist, sollte der Decodierungsprozess anhalten (nachdem alle anderen in der Szenenendmarke enthaltenen Anweisungen implementiert wurde).

scene_reveal_flag (Flag Szene zeigen)

Wenn dieses 1-Bit-Flag auf '1' gesetzt ist, weist es den Decoder an, das unmittelbar vorangegangene Seitenaufbausegment zu implementieren. Das heißt, die Anzeige sollte von dem im Augenblick aktiven Seitenaufbausegment zu dem Seitenaufbausegment wechseln, das davor decodiert wurde. Wenn dieses Flag auf '0' gesetzt ist, wird die Implementierung des Seitenaufbausegmentes verschoben, bis ein 'Szene zeigen' mit dem passenden scene_tag gesendet wird.

send_scene_done (Szene abgeschlossen senden)

Wenn dieses 1-Bit-Flag auf '1' gesetzt ist, weist es den Decoder an eine APDU 'Szene abgeschlossen' an die Anwendung zu senden.

scene_tag (Szene-Tag)

Diese (ganzzahlige) 4-Bit-Zahl wird durch die Anwendung gesetzt. Ihr Wert sollte für jede Szenenendmarke modulo 16 weiterrücken.

Nachricht Szene abgeschlossen (scene_done_message)

Tabelle 41 – Codierung der Nachricht Szene abgeschlossen

Syntax	Anzahl der Bits	Mnemonik
scene_done_message() {		
scene_done_message_tag	**24**	**uimsbf**
length_field()		
decoder_continue_flag	**1**	**bslbf**
scene_reveal_flag	**1**	**bslbf**
reserved	**2**	**bslbf**
scene_tag	**4**	**uimsbf**
}		

Der Decoder sollte eine 'Nachricht Szene abgeschlossen' senden, wenn er die Decodierung eines Anzeigesatzes gefolgt von einer Szenenendmarke mit einem gesetzten Flag 'Szene abgeschlossen senden' abschließt.

decoder_continue_flag (Flag Decodieren fortsetzen)

scene_reveal_flag (Flag Szene zeigen)

Diese 1-Bit-Flags wiederholen den Zustand desselben Flags in der Szenenendmarke, die das Senden der Nachricht verursachte.

scene tag (Szene-Tag)

Diese (ganzzahlige) 4-Bit-Zahl wiederholt den Zustand derselben Zahl in der Szenenendmarke, die das Senden der Nachricht verursachte.

Szenen-Steuerung (scene_control)

Tabelle 42 – Codierung von Szenen-Steuerung

Syntax	Anzahl der Bits	Mnemonik
scene_control() {		
scene_control_tag	**24**	**uimsbf**
length_field()		
decoder_continue_flag	**1**	**bslbf**
scene_reveal_flag	**1**	**bslbf**
reserved	**2**	**bslbf**
scene_tag	**4**	**uimsbf**
}		

Die Anwendung kann eine APDU-Szenensteuerung nur senden, nachdem die 'Nachricht Szene abgeschlossen' für den entsprechenden Anzeigesatz von dem Decoder gesendet wurde.

decoder_continue_flag (Flag Decodieren fortsetzen)

Wenn dieses 1-Bit-Flag auf '1' gesetzt ist, weist es den Decoder an, die Decodierung der Untertiteldaten fortzusetzen. Dies hat nur einen Effekt, wenn das Flag Decodierung fortsetzen in der Szenenendmarke auf '0' gesetzt war.

scene_reveal_flag (Flag Szene zeigen)

Wenn dieses 1-Bit-Flag auf '1' gesetzt ist, weist es den Decoder an, die Implementierung des neuen Seitenaufbau-Segmentes für diese Szenen-Untertiteldaten fortzuführen. Dies hat nur eine Wirkung, wenn das Flag Szene zeigen in der Szenenendmarke auf '0' gesetzt war.

8.6.4.5 Objekt Herunterladen

Die 'Untertitel-Herunterladen'-APDU ist im Format mit der Untertitelsegment-APDU identisch. Die APDU ist darauf beschränkt, dass sie nur Datensegmente der DVB-Untertitelungsobjekte übertragen sollte (wobei die Untertitelsegment-APDU jedes DVB-Untertitelungsegment übertragen kann).

Es gibt verschiedene Wahlmöglichkeiten für die Implementierung des Speicherns der Objekte im Cache. Die der Anwendung zur Verfügung gestellte Funktion entspricht Folgendem:

– Das durch die APDU übertragene Datensegment des DVB-Untertitelungsobjektes wird in dem Objektcache gespeichert.

– Wenn ein Bereich auf ein Objekt in dem Cache verweist (mit entsprechender Objekt-ID und Objektversorger-Flag), wird das DVB-Untertitelungssegment aus dem Cache gelesen und dem Eingang des Decoders zugeleitet.

Die Worst-Case-Geschwindigkeit des Decoders ist mit der identisch, mit der, wenn es gefordert wird, das Segment über die Einheitliche Schnittstelle weitergeleitet wird. Die Verarbeitungsbelastung wird jedoch an der Anwendung reduziert und kann am Decoder reduziert werden.

Alle Objekttypen, die durch DVB-Untertitelung erkannt werden, können im Cache gespeichert werden. Wenn jedoch der Objekttyp ein Zeichen oder eine Zeichenfolge ist, dann wird die Darstellung des einzelnen Zeichens am Bildschirm in dieser Norm nicht angesprochen.

Die 'Untertitel herunterladen'-APDU kann zu jeder Zeit gesendet werden, so dass eine Untertitelsegment-APDU ein Objekt-Datensegment übertragen könnte.

Wenn die MMI-Sitzung eröffnet ist, wird vorausgesetzt, dass keine Daten im Cache sind. Wenn die MMI-Sitzung geschlossen wird, wird der Inhalt des Cache gelöscht. Das Objekt Cache Räumen ermöglicht auch, den Cache zu löschen.

Tabelle 43 – Codierung von Untertitel Herunterladen (subtitle_download)

Syntax	Anzahl der Bits	Mnemonik
subtitle_download() {		
subtitle_download_tag	**24**	**uimsbf**
length_field()		
DVB_Subtitling_segment()		
}		

Tabelle 44 – Codierung von Herunterladedaten Entfernen (flush_download)

Syntax	Anzahl der Bits	Mnemonik
flush_download() {		
flush_download_tag	**24**	**uimsbf**
length_field()		
}		

Fordert an, dass der Untertitelungs-Objekt-Cache des Decoders gelöscht wird.

Tabelle 45 – Codierung Herunterladen Antwort (download_reply)

Syntax	Anzahl der Bits	Mnemonik
download_reply() {		
download_reply_tag	**24**	**uimsbf**
length_field()		
objekt_id	**16**	**uimsbf**
download_reply_id	**8**	**uimsbf**
}		

Das Objekt Herunterladen Antwort ermöglicht dem Hauptgerät, Probleme mit dem Herunterladen eines Objektes anzuzeigen. Wenn das Herunterladen erfolgreich war, gibt es keine Notwendigkeit für eine Antwort.

object_id (Objekt-Kennung)
Die Kennung des heruntergeladenen Objektes, das die Nachricht verursachte. Dort teilt die Nachricht mit, dass das Segment kein Objekt-Datensegment war, die object_id sollte 0xFFFF sein.

download_reply_id	
Herunterladen OK	00
Kein Objekt-Datensegment	01
Speicher gelöscht	02
Reserviert	Andere Werte

8.6.4.6 Von der einheitlichen Schnittstelle adressiertes Untertitelungs-Decodermodell

Es bringt keinen Anstieg in der geforderten Verarbeitungsgeschwindigkeit. Die Schnittstelle ist jedoch in der Lage, eine gegebenenfalls verfügbare höhere Verarbeitungsgeschwindigkeit auszunutzen.

Bei Betrieb im Overlaymode sollte der Untertitelungsdecoder denselben Pixel-Puffer und denselben Seitenaufbau-Puffer bereitstellen wie in [9] definiert ist. Wenn die Anwendung jedoch von der einheitlichen Schnittstelle adressiert wird, wird sie mit genügend Information über die Eigenschaften des Anzeigesystems versorgt, um sicherzustellen zu können, dass keine unvorhergesehene Farbtiefenquantisierung auftritt. Auch alle Hilfsmittel des Hauptgerätes, die über die von dem Minimalspezifikation-DVB-Untertitelungsdecoder geforderten hinausgehen, können durch die Anwendung ausgenutzt werden.

Der Graphikmode ganzer Bildschirm wird durch die DVB-Untertitelungs-Spezifikation nicht angesprochen. Wenn diese Betriebsart unterstützt wird, schreibt diese Spezifikation folgende Hauptgerätehilfsmittel vor:

Mindest-Pixel-Pufferspeicher	207360 Bytes (im Prinzip genügen 720 × 576 × 16 Farben)
Mindest-Seitenaufbau-Pufferspeicher	12 kBytes

Für das Objekt-Cache-RAM gibt es keine verbindliche Mindestanforderung. Wenn das Anzeigeprofil anzeigt, dass kein Objekt-Cache-Ram vorgesehen ist, dann sollte die Einrichtung zum Herunterladen eines Objektes von einer Anwendung nicht benutzt werden.

Benutzen von Objektcache

Objekte 'Bitmap Herunterladen' werden durch Einbeziehen ihrer Objekt-ID (object_id) und eines Objektversorger-Flags (object_provider_flag) = 0x01 in das Bereichsaufbau-Segment (das anzeigt, dass der Objektversorger ein residentes ROM im Hauptgerät ist) gebildet.

Die Worst-Case-Geschwindigkeit des Decoders ist mit der identisch, mit der das Segment, wenn es gefordert wird, über die Einheitliche Schnittstelle weitergeleitet wird. Die Arbeitsbelastung an der Anwendung wird aber reduziert und die Arbeitsbelastung am Decoder kann durch Benutzen des Cache reduziert werden. So bietet dieses Verfahren die Gelegenheit, die Leistung des Systems zu verbessern.

Eine Einrichtung zum Zwischenspeichern von Objekten wird empfohlen, ist aber nicht obligatorisch.

Seiten-ID

Die Verwendung der einheitlichen Schnittstelle bei DVB-Untertitelung erfordert, dass der Decoder sich verhält als ob die DVB-Untertitelsegmente an dem Eingang des Blocks 'Untertitel-Verarbeitung' nach dem Puffer der codierten Daten des Untertiteldecoders eingesetzt werden. So ist dieser Einfügepunkt hinter dem Seitenfilter, und deshalb kann die Seiten-ID des DVB-Untertitelungssegmentes durch den Decoder ignoriert werden und braucht nicht bei der Anwendung eingegeben zu werden.

Weitere Untertitelungsdienste

Für den Decoder wird nur vorausgesetzt, dass er einen einzigen Untertiteldecoder hat. Wenn der Decoder bereits einen Rundfunk-Untertitelstrom decodiert, wenn die MMI-Sitzung beginnt, kann der Decoder den Decodierungsbetrieb der Rundfunkdaten für die Dauer der MMI-Sitzung aussetzen. Nachdem die Sitzung geschlossen ist, sollte der Decoder, wenn möglich, zu seiner vorhergehenden Decodierungsaufgabe zurückkehren.

8.6.5 MMI-Objekte auf höherer Ebene

Die nachstehenden Objekte werden nur im MMI-Mode auf höherer Ebene benutzt.

Die MMI-Objekte auf höherer Ebene definieren die erforderlichen Dialoge, erlauben aber dem Hauptgerät, das Format und den Typ der Anzeige zu bestimmen. Die Anwendung kann im Text Steuerzeichen benutzen, sie dürfen aber vom Hauptgerät ignoriert werden. Die Steuerzeichen können vom Hauptgerät dazu benutzt werden, bei der Darstellung des Menüs zu helfen.

Der Hauptteil jedes Textes wird nach dem aktuellen Zeichensatz interpretiert. Die Darstellung der Information durch das Hauptgerät ist nicht auf die Benutzung einer Anzeigeeinrichtung beschränkt.

8.6.5.1 Text (text)

Das Objekt Text wird benutzt, um einen Textblock auf dem Bildschirm anzuzeigen. Es wird als Teil eines Objektes auf höherer Ebene im MMI-Mode auf hoher Ebene benutzt.

Tabelle 46 – Codierung des Objektes Text

Syntax	Anzahl der Bits	Mnemonik
text() {		
text_tag	**24**	**uimsbf**
length_field()		
for (i=0;i<length;i++)		
text_char	**8**	**uimsbf**
}		

Ein Objekt Text mit text_length gleich 0 wird als Nullobjekt interpretiert: es wird nichts angezeigt.

text_char (Textzeichen)

Textinformation wird unter Verwendung der in [4] beschriebenen Zeichensätze und Verfahren codiert.

Der von der Anwendung geschickte Text darf Steuerzeichen enthalten, wie sie in [4] definiert sind, und die Hinweise liefern, wie die Anzeige darzustellen ist. Die Darstellung dieser Steuerzeichen hängt vom Hauptgerät ab.

8.6.5.2 Anfrage (enq)

Mit Enq und Answ (Antwort) kann die Anwendung eine Benutzereingabe anfordern. Es gibt eine einzige Anweisung, die eine Anfrage nach Information wie z. B. der Identifizierungsnummer des Benutzers sein kann. Die Antwort auf Enq wird in dem Objekt Answ des Hauptgeräts gegeben. Das Objekt Enq ermöglicht der Anwendung festzulegen, ob die Benutzereingabe durch das Hauptgerät dem Benutzer rückgemeldet werden sollte. Bei der Eingabe einer Identifizierungsnummer sollten z. B. die vom Benutzer eingegebenen Ziffern nicht angezeigt werden.

Tabelle 47 – Codierung des Objektes Enq

Syntax	Anzahl der Bits	Mnemonik
enq () {		
enq_tag	**24**	**uimsbf**
length_field()		
reserved	**7**	**bslbf**
blind_answer	**1**	**bslbf**
answer_text_length	**8**	**uimsbf**
for (i=0;i<enq_length-2;i++)		
text_char	**8**	**uimsbf**
}		

blind_answer ('blinde' Antwort):

Auf 1 gesetzt bedeutet, dass die Benutzereingabe beim Eingeben nicht angezeigt wird. Das Hauptgerät kann die verwendeten Ersatzzeichen auswählen (Stern, ...).

answer_text_length (Antwort-Textlänge):

Erwartete Länge der Antwort. Auf hex 'FF' setzen, wenn der Anwendung unbekannt.

text_char (Textzeichen)

Textinformation wird unter Verwendung der in [4] beschriebenen Zeichensätze und Verfahren codiert.

Beispiel von Enq:

PLEASE TYPE YOUR PIN CODE	/* Text der Anfrage */
'****'	/*blinde Antwort*/

8.6.5.3 Antwort (answ)

Dieses Objekt wird zusammen mit dem Objekt Enq benutzt, um Benutzereingaben abzuwickeln.

Tabelle 48 – Codierung des Objektes Answ

Syntax	Anzahl der Bits	Mnemonik
answ() {		
answ_tag	**24**	**uimsbf**
length_field()		
answ_id	**8**	**uimsbf**
if (answ_id == answer) {		
for (i=0;i<length;i++)		
text_char	**8**	**uimsbf**
}		
}		

Wenn die Anwendung den Benutzer zu einer Eingabe auffordert, sendet sie das Objekt Enq. Das Hauptgerät antwortet mit dem Objekt Answ (nach einer möglichen Übersetzung des Tastendruck-Codes in char_value-Codes). Die text_chars in dem Antwortobjekt müssen nach demselben Schema für die Zeichencodierung wie in dem zugehörigen Objekt Enq codiert werden. Das Textfeld in dem Objekt Answ muss dieselbe Zeichengabe (in [4] definiert) für die Auswahl der Zeichencodetabellen anwenden wie in dem dazugehörenden Objekt Enq.

Das answ_id-Feld dient zur Anzeige des Typs der Antwort, die vom Benutzer empfangen wurde. answ_id-Werte werden wie folgt codiert:

answ_id		Wert
cancel	Löschen	00
answer	Antwort	01
reserved	Reserviert	Andere Werte

Der Wert 'Antwort' bedeutet, dass das Objekt die Eingabe des Benutzers enthält (die die Länge 0 haben kann). Der Wert 'Löschen' bedeutet, dass der Benutzer den Dialog abbrechen will.

8.6.5.4 Menü (menu)

Dieses Objekt wird in Verbindung mit dem Objekt Menüantwort benutzt, um die Menüs im MMI-Mode auf hoher Ebene zu verwalten.

Tabelle 49 – Codierung des Objektes Menü

Syntax		Anzahl der Bits	Mnemonik
menu () {			
menu_tag		**24**	**uimsbf**
length_field()			
choice_nb		**8**	**uimsbf**
TEXT()	/* Titeltext*/		
TEXT()	/* Untertiteltext*/		
TEXT()	/* Fußzeilentext */		
for (i=0;i<choice_nb;i++) {	/* when choice_nb _'FF' */		
TEXT()			
}			

Ein Menü besteht aus einem Titel, einem Untertitel, mehreren Auswahltexten und einer Fußzeile. Textobjekte mit text_length = 0 können benutzt werden (wenn z. B. kein Untertitel oder keine Fußzeile benutzt wird).

Choice_nb = 'FF' bedeutet, dass dieses Feld nicht die Nummer der gewählten Information überträgt.

Die Art wie das Hauptgerät den Text und den Menürahmen anzeigt und die Auswahltexte auswählt, hängt vom Hersteller ab. Das Hauptgerät kann z. B. die Auswahl auf mehreren Seiten anzeigen und die 'page-down'- und 'page-up'-Funktion selbst durchführen. Dem Hersteller ist auch freigestellt zu definieren, wie der Benutzer die Auswahl vornimmt (numerisches Tastenfeld, Pfeiltasten, farbige Tasten, Stimme ...).

8.6.5.5 Menüantwort (menu answ)

Dieses Objekt wird in Verbindung mit dem Objekt Menu benutzt, um mit der vom Benutzer durchgeführten Auswahl zu antworten und mit dem Objekt Liste anzuzeigen, dass der Benutzer dieses Objekt beendet hat.

Tabelle 50 – Codierung des Objektes Menüantwort

Syntax	Anzahl der Bits	Mnemonik
menu_answ () {		
menu_answ_tag	**24**	**uimsbf**
length_field()=1		
choice_ref	**8**	**uimsbf**
}		

choice_ref (Auswahl-Bezugsnummer):

Die Nummer des vom Benutzer gewählten Auswahltextes. Wenn dem Objekt das Objekt Menü vorausgegangen war, dann entspricht choice_ref der ersten Auswahl, die in diesem Objekt von der Anwendung dargestellt wurde (erster Auswahltext nach dem Fußzeilentext in dem Objekt Menü) und choice_ref = 2 entspricht dem zweiten von der Anwendung dargestellten Auswahltext.

choice_ref = 00 zeigt an, dass der Benutzer das vorhergehende Menü oder das Objekt Menü gelöscht hat, ohne eine Auswahl vorzunehmen.

Beispiel eines Menüs:

DO YOU WANT TO BUY	/* Titel */
"JURASSIC PARK" COST: $5.00	/* Untertitel */
1/ YES	/* Auswahltext 1 */
2/ NO	/* Auswahltext 2 */
< Remaining Credit : $47.50 >	/* Fußzeile */
< The FIRST 3 MINUTES ARE FREE OF CHARGE >	

8.6.5.6 Liste (list)

Dieses Objekt wird benutzt, um eine Liste der anzuzeigenden Elemente zu senden (z. B. während eines Dialogs über die Berechtigungen). Sie hat genau dieselbe Syntax wie das Objekt Menü und wird in Verbindung mit dem Objekt Menu_Answ benutzt.

Tabelle 51 – Codierung des Objektes Liste

Syntax		Anzahl der Bits	Mnemonik
list () {			
list_tag		**24**	**uimsbf**
length_field()			
item_nb		**8**	**uimsbf**
TEXT()	/* Titeltext */		
TEXT()	/* Untertiteltext */		
TEXT()	/* Fußzeilentext */		
for (i=0;i<item_nb;i++) {	/* when item_nb _'FF' */		
TEXT()			
}			

Eine Liste besteht aus einem Titel, einem Untertitel, einigen Elementen und einer Fußzeile. Textobjekte mit der text_length = 0 können benutzt werden (z. B. wenn kein Untertitel oder keine Fußzeile benutzt wird).

Item_nb = 'FF' bedeutet, dass dieses Feld nicht die Information über die Anzahl der Elemente überträgt.

Die Art wie das Hauptgerät den Text und den Menürahmen anzeigt und die Elemente auswählt, hängt vom Hersteller ab. Z. B. kann das Hauptgerät die Elemente auf mehreren Seiten anzeigen und die 'page-down'- und 'page-up'-Funktion selbst durchführen.

8.7 Kommunikationshilfsmittel

8.7.1 Hilfsmittelklasse Kommunikation mit niedriger Datenrate

8.7.1.1 Einführung

Ein Teil des Hauptgerätes ist eine Schnittstelle der Hilfsmittelklasse 'Kommunikation mit geringer Datenrate'. Sie stellt eine bidirektionale Kommunikation z. B. über eine Telefonleitung oder einen Kabelnetz-Rückkanal zur Verfügung. Sie kann in Verbindung mit interaktiven Diensten benutzt werden und Funktionen für die Zugriffsbeschränkung unterstützen.

Die Hilfsmittelklasse wird in allgemeiner Form definiert, so dass verschiedene zugrunde liegende Kommunikationstechnologien gemeinsam benutzt werden können. Innerhalb der Klasse werden Hilfsmitteltypen definiert, um die Kommunikation mit einzelnen Baueinheitentypen zu unterstützen, die den gemeinsamen Satz der Objekte benutzen. Das Modell ist ein gleichzeitig bidirektionaler Kanal (voll Duplex), über den Kommunikation von beliebigen Daten möglich ist. Die Ablaufsteuerung wird in beiden Richtungen zwischen Anwendung und Hauptgerät angewendet. Zur Übermittlung werden die Daten in Segmente aufgeteilt, um die Größe der Puffer zu begrenzen, die sowohl in der Anwendung als auch im Hauptgerät benötigt werden. Das Protokoll der Ablaufsteuerung begrenzt auch die Anzahl der Puffer, die sowohl in der Anwendung als auch im Hauptgerät benötigt werden.

Die nachstehend beschriebenen APDUs werden bei Funktionen der Kommunikation mit geringer Datenrate vom Hauptgerät und der Anwendung benutzt.

8.7.1.2 Anforderungen

1. Puffer von bis zu 254 Bytes Länge müssen in beiden Richtungen akzeptiert werden. Die Anwendung kann, wenn erforderlich, eine kleinere maximale Puffergröße auswählen.

2. Zwischen Hauptgerät und Anwendung wird eine Ablaufsteuerung implementiert. Das Hauptgerät muss einen zweiten Puffer für die Übertragung akzeptieren, solange der erste gesendet wird, ein dritter muss jedoch warten bis der erste übertragen wurde. Ein ähnliches Protokoll wird für die von der Kommunikationsschnittstelle ankommenden Daten benutzt. Das Hauptgerät ist dafür zuständig, an den eingerichteten Verbindungen über externe Übertragungsabschnitte eine Ablaufsteuerung einzusetzen und mit der Ablaufsteuerung zwischen Anwendung und Hauptgerät zu koordinieren.

8.7.1.3 Objekte, die das Hilfsmittel Kommunikation mit niedriger Datenrate unterstützen

Es werden vier Objekte definiert: Comms Cmd (Kommunikationsbefehl), Comms Reply (Kommunikationsantwort), Comms send (Kommunikation Senden) und Comms Rcv (Kommunikation Empfangen). Comms Cmd wird von der Anwendung gesendet und erlaubt mehrere Verwaltungsvorgänge an dem auszuführenden Kommunikationshilfsmittel. Comms Reply wird vom Hauptgerät gesendet und bestätigt einen Comms Cmd. Comms Reply kann auch ein Comms send bestätigen, das die abgehende Ablaufsteuerung zu dem Kommunikationshilfsmittel durchführt. Comms send ist ein Puffer mit Daten, die zu dem Kommunikationshilfsmittel gesendet werden. Dies sind über die Leitung abzusendende Daten. Comms Rcv ist ein Puffer mit Daten, die von dem Kommunikationshilfsmittel ausgegeben werden. Dies sind von der Leitung zu empfangende Daten. Vorher durch ein Objekt Comms Cmd gesetzte Puffergröße und Zeitbegrenzungen gelten für empfangene Puffer.

Um ein einzelnes Kommunikationshilfsmittel zu benutzen, muss zu dem Hilfsmittel eine Sitzung eingerichtet werden, wobei der Standardmechanismus zum Einrichten einer Sitzung benutzt wird. Transaktionen, die die Kommunikationsobjekte benutzen, finden innerhalb dieser Sitzung statt.

8.7.1.4 Kommunikationsbefehl (Comms Cmd)

Tabelle 52 – Codierung des Objektes Kommunikationsbefehl

Syntax	Anzahl der Bits	Mnemonik
comms_cmd() {		
comms_cmd_tag	**24**	**uimsbf**
length_field()		
comms_command_id	**8**	**uimsbf**
if (comms_command_id = = Connect_on_Channel) {		
connection descriptor()		
retry_count	**8**	**uimsbf**
timeout	**8**	**uimsbf**
}		
if (comms_command_id = = Set_Params) {		
buffer_size	**8**	**uimsbf**
timeout	**8**	**uimsbf**
}		
if (comms_command_id = = Get_Next_Buffer) {		
comms_phase_id	**8**	**uimsbf**
}		
}		

Tabelle 53 – Codierung des Objektes Verbindungsbeschreibung

Syntax	Anzahl der Bits	Mnemonik
connection_descriptor() {		
connection_descriptor_tag	**24**	**uimsbf**
length_field()		
connection_descriptor_type	**8**	**uimsbf**
if (connection_desciptor_type == SI_Telephone_Descriptor) {		
telephone_descriptor() /* siehe DVB/SI-Specifikation [4] */		
}		
if (connection_descriptor_type == Cable_Return_Channel_Descriptor) {		
channel_id	**8**	**uimsbf**
}		
}		

comms_command_id	Kommunikationsbefehl-Kennung	id-Wert
Connect_on_Channel	Mit Kanal verbinden	01
Disconnect_on_Channel	Von Kanal trennen	02
Set_Params	Parameter einstellen	03
Enquire_Status	Status anfordern	04
Get_Next_Buffer	Nächsten Puffer holen	05
reserved	Reserviert	Weitere Werte

connection_desriptor_type	Typ der Verbindungsbeschreibung	Typ-Wert
SI_Telephone_Descriptor	SI-Telefon-Beschreibung	01
Cable_Return_Channel_Descriptor	Rückkanal-Beschreibung	02
reserved	Reserviert	Weitere Werte

Connect_on_Channel (Mit Kanal verbinden) richtet eine Kommunikation mit dem Kommunikationshilfsmittel ein. Die connection_descriptor (Verbindungsbeschreibung) enthält die Information, die zum Einrichten der Verbindung benötigt wird, z. B. beim Benutzen eines Modems die zu wählende Telefonnummer. retry_count (Wiederholung zählen) erlaubt einen oder mehrere erneute Versuche. Mit timeout (Zeitbegrenzung) in Sekunden kann ein Verbindungsversuch abgebrochen werden, wenn innerhalb einer bestimmten Zeit keine positive Anzeige des Zustandes der Verbindung empfangen wird. Ein Wert für die Zeitbegrenzung gleich Null bedeutet unbestimmtes Warten.

Disconnect_on_Channel (vom Kanal trennen) beendet die Verbindung über das Kommunikationshilfsmittel.

Set_Params (Parameter einstellen) überträgt zwei Parameter. Der erste ist ein 8-Bit-Wert, der die maximale Puffergröße in Bytes angibt, die auf mindestens 1 und höchstens 254 beschränkt ist. Der zweite Parameter ist ein 8-Bit-Wert, der eine Eingabezeitbegrenzung in Einheiten von $10\ \mathrm{ms}$ angibt. Wenn ein oder mehrere Bytes in dem aktuellen Puffer empfangen wurden, und dann eine Zeit gleich der Zeitbegrenzung verstrichen ist, ohne dass weitere Daten empfangen wurden, wird der Puffer als Objekt Comms Rcv an die Anwendung gegeben. Wenn der Puffer bis zu dem durch den Parameter buffer_size festgelegten Grenzwert ohne Zeitbegrenzung gefüllt ist, dann wird der Puffer unmittelbar zurückgegeben.

Enquire_Status (Status anfragen) hat keine Parameter und erzeugt ein Objekt Comms Reply mit einem Parameter, der den augenblicklichen Zustand der Kommunikationsverbindung anzeigt.

Get_Next_Buffer (nächsten Puffer holen) führt das Protokoll der Ablaufsteuerung auf der Empfängerseite durch. Nachdem auf dem Kanal eine Verbindung erfolgreich eingerichtet wurde, gibt die Anwendung ein Objekt Get_Next_Buffer mit auf 0 gesetzter comms_phase_id (Kommunikationsphasen-Kennung) aus. Die erste comms_phase_id muss 0 sein und das Hauptgerät muss dies durchführen. Die Anwendung kann dann auch Get_Next_Buffer mit auf 1 gesetzter comms_phase_id ausgeben. Dann kann sie so lange keine weiteren Daten ausgeben, bis ein Datenpuffer auf dem Kanal empfangen und unter Benutzung des Objektes Comms Rcv zu der Anwendung geschickt wurde. Erst dann kann sie Get_Next_Buffer mit wieder auf 0 gesetzter comms_phase_id ausgeben. Die comms_phase_id wird mit abwechselnden Werten 0, 1, 0, 1 usw. verwendet. Damit steuert die Anwendung die Datenrate, mit der das Hauptgerät ihr Daten senden kann. Der Phasenwert erlaubt der Anwendung, eindeutig zu identifizieren, welcher Puffer gesendet werden muss. Die Ausgabe von Get_Next_Buffer fordert eine unmittelbare Bestätigung Kommunikationsantwort, und der empfangene Puffer wird unter Benutzung des Objektes Comms Rcv irgendwann später gesendet.

8.7.1.5 Kommunikationsantwort (comms reply)

Tabelle 54 – Codierung des Objektes Kommunikationsantwort

Syntax	**Anzahl der Bits**	**Mnemonik**
comms_reply() {		
comms_reply_tag	**24**	**uimsbf**
length_field()		
comms_reply_id	**8**	**uimsbf**
return_value	**8**	**uimsbf**
}		

comms_reply_id	Kommunikations-Antw.-Kennung	id-Wert
Connect_Ack	Verbinden bestätigen	01
Disconnect_Ack	Trennung bestätigen	02
Set_Params_Ack	Parameter einstellen bestätigen	03
Status_Reply	Status antworten	04
Get_Next_Buffer_Ack	Nächsten Puffer holen bestätigen	05
Send_Ack	Senden bestätigen	06
reserved	Reserviert	Weitere Werte

Im Allgemeinen sind positive Antwortwerte richtig und negative Antwortwerte sind Fehler. 0 ist der Standard-Antwortwert für richtig und -1 ist der unspezifische Fehler. Weitere Fehlerwerte sind noch zu definieren. Der Antwortwert für Status_Reply hat die Werte 0 = nicht verbunden und 1 = verbunden.

Der Antwortwert für Send_Ack (Bestätigung Senden) teilt mit, welcher Puffer erfolgreich gesendet wurde. Alle über die Kommunikation gesendeten Puffer haben eine Phase 0 oder 1. 0 ist die nach einem erfolgreichen Kommunikationsbefehl 'Connect' als erste gesendete Phase. Nachfolgende Puffer haben abwechselnde Phase 1, 0, 1, 0 usw. Wenn ein Puffer mit Phase 0 durch ein Send_Ack mit Antwortwert 0 bestätigt wird, kann der nächste Puffer mit Phase 0 gesendet werden. Ähnlich bestätigt ein Send_Ack mit einem Wert 1 den vorangegangenen Puffer mit Phase 1 und der nächste Puffer mit Phase 1 kann nun gesendet werden. Dieser Mechanismus steuert den Datenfluss von der Anwendung zu der Kommunikation und fordert das Hauptgerät auf, nur zwei Puffer zur Verfügung zu stellen – einen aktuell gesendeten und den nächsten.

Wenn sich ein Fehler ereignet, kann dieses Objekt unaufgefordert an das Modul gesendet werden. Der einzige allgemein signalisierte Fehler ist eine Unterbrechung, bei der das Hauptgerät comms_reply mit comms_reply_id sendet, die auf Status_Reply und einen Antwortwert von 0 (bedeutet 'unterbrechen') gesetzt ist.

8.7.1.6 Kommunikation Senden (Comms send)

Tabelle 55 – Codierung des Objektes Kommunikation Senden

Syntax	**Anzahl der Bits**	**Mnemonik**
comms_send() {		
comms_send_tap	**24**	**uimsbf**
length_field()		
comms_phase_id	**8**	**uimsbf**
for (i=0;i<n;i++) {		
message_byte	**8**	**uimbf**
}		
}		

Die comms_phase_id nimmt die Werte 0 oder 1 an. Das erste Comms Send (Kommunikation Senden) nach einem Kommunikationsbefehl 'Connect' muss comms_phase_id auf 0 setzen. Nachfolgende Comms sends werden in abwechselnder Folge auf die Werte 1, 0, 1, 0 usw. gesetzt. Das Hauptgerät überprüft dies durch Beantworten mit einer Comms Reply (Kommunikationsantwort) mit einem Send_Ack-Fehler, wenn die Sequenz abgebrochen ist. Dies erlaubt der Anwendung eindeutig zu bestimmen, welcher von den gesendeten Puffern bestätigt wurde. Die Komplexität, zwei Phasen zu haben anstelle eines einfachen Senden-Bestätigen-Senden-Bestätigen-Mechanismus, ermöglicht dem Hauptgerät, die Kommunikation fortlaufend mit Daten versorgt zu halten, so dass es mit maximaler Geschwindigkeit arbeiten kann.

Eine Nachricht besteht aus maximal 254 Bytes.

8.7.1.7 Kommunikation Empfangen (Comms Rcv)

Tabelle 56 – Codierung des Objektes Kommunikation Empfangen

Syntax	Anzahl der Bits	Mnemonik
comms_rcv() {		
comms_rcv_tag	**24**	**uimsbf**
length_field()		
comms_phase_id	**8**	**uimsbf**
for (i=0;i<n;i++) {		
message_byte	**8**	**uimbf**
}		
}		

Comms_phase_id zeigt an, zu welcher Phase des Get_Next_Buffer-Zyklus diese Daten gehören.

8.8 Hilfsmittelkennungen und Anwendungsobjekt-Tags

8.8.1 Hilfsmittelkennungen

Kennungen für die öffentlichen Hilfsmittel, die in dieser Spezifikation definiert werden, sind in folgender Tabelle angegeben.

Tabelle 57 – Werte für Hilfsmittelkennung

Hilfsmittel	Klasse	Typ	Version	Hilfsmittel-Kennung
Hilfsmittelmanager	1	1	1	00010041
Anwendungsinformation	2	1	1	00020041
Zugriffsbeschränkungs-Unterstützung	3	1	1	00030041
Hauptgerätesteuerung	32	1	1	00200041
Datum/Zeit	36	1	1	00240041
MMI	64	1	1	00400041
Kommunikation niedriger Datenrate	96	siehe 8.8.1.1	1	0060xxx1
Reserviert	Weitere Werte	Weitere Werte	Weitere Werte	Weitere Werte

8.8.1.1 Hilfsmitteltypen für Kommunikation niedriger Datenrate

Der Wert des Hilfsmitteltyps für Kommunikation niedriger Datenrate codieren zwei Felder. Das erste Feld, das die Bits 0 und 1 des Hilfsmitteltyps benutzt, ist eine Baueinheitennummer. Es können mehrere der einzelnen Baueinheiten vorkommen, z. B. von einem Modem. Das Baueinheiten-Nummernfeld identifiziert welcher. Das zweite Feld mit den Bits 2 bis 9 ist der genaue Baueinheitentyp. Für bestimmte Baueinheiten wie Modems wird das Feld in zwei Teilfelder aufgeteilt.

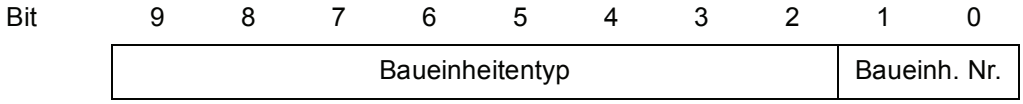

Bild 14 – Struktur des Hilfsmitteltyps für Kommunikation mit niedriger Datenrate

Das Feld des Baueinheitentyps wird wie folgt codiert:

Beschreibung	Wert
Modems – siehe unten	00 bis 3F
Serieller Anschluss	40 bis 4F
Kabel-Rückkanal	50
Reserviert	51 bis FF

Das Feld des Hilfsmitteltyps für Modems wird in drei Felder codiert. Die Baueinheitennummer ist dieselbe wie oben, der Baueinheitentyp ist jedoch in ein Feld aufgeteilt, das für die erforderliche Datenverarbeitung codiert ist, und in ein Feld, das den Modemtyp angibt und nach der Spezifikation der Übertragungsgeschwindigkeit codiert ist, die die ITU-TV-Serien-Nummern benutzt.

Bit	9	8	7	6	5	4	3	2	1	0
	0	0	Modemtyp				Datenverarb.		Baueinh.-Nr.	

Bild 15 – Struktur des Hilfsmitteltyps für Modems

Das Feld 'Datenverarbeitung' wird wie folgt codiert

Beschreibung	Wert
Verhandeln	00
Nr. V.24bis	01
Keine Datenverarbeitung	10
Reserviert	11

'Verhandeln' bedeutet, sowohl die V.42-Fehlerkorrektur als auch die V.42bis-Datenkompression zu benutzen, wenn dies mit dem anderen Ende (der Leitung) ausgehandelt werden kann. 'Nr. V.42bis' bedeutet verhandeln, um die V.42-Fehlerkorrektur zu benutzen, aber nicht die V.42bis-Datenkompression.

Das Feld für den Modemtyp wird wie folgt codiert:

Beschreibung	Wert
Reserviert	0000
Reserviert	0001
V.21 (300 bit/s)	0010
Reserviert	0011
V.22 (1 200 bit/s)	0100
V.22bis (2 400 bit/s)	0101
V.23 (1 200/75 bit/s)	0110
Reserviert	0111
V.32 (9 600/4 800 bit/s)	1000
V.32bis (14,4 kbit/s)	1001
V.34 (28,8 kbit/s)	1010
Reserviert	1011 bis 1111
V.27ter	1110
V.29	1111

Wenn das spezielle vom Hauptgerät benutzte Modem Mehrfachübertragungsraten der Serie V bietet, dann muss das Hauptgerät alle verfügbaren Geschwindigkeiten als verschiedene Hilfsmitteltypen anbieten. Wenn ein spezieller Hilfsmitteltyp in Gebrauch ist, dann müssen alle anderen von diesem Modem angebotenen Übertragungsraten auch belegt werden.

8.8.2 Anwendungsobjekt-Tags

Die Codierung des apdu-tag erfolgt nach den ASN.1-Regeln. Jedes apdu_tag wird in 3 Bytes codiert. Unter den 24 Bits jedes apdu_tags sind 10 durch die ASN.1-Regeln feststehend, wie es im folgenden Bild gezeigt wird. Es werden nur einfache Tags benutzt.

Byte 1 b24 b17	**Byte 2** b16 b9	**Byte 3** b8 b1
1 0 0 1 1 1 1 1	1 x x x x x x x	0 x x x x x x x

Bild 16 – Codierung mit einfachen Tags

Tabelle 58 – Tag-Werte der Anwendungsobjekte

apdu-Tag	Tag-Wert (hex)	Hilfsmittel	Richtung Hauptgerät ↔ Anwendung
$T_{profile\_enq}$	9F 80 10	Hilfsmittelmanager	↔
$T_{profile}$	9F 80 11	Hilfsmittelmanager	↔
$T_{profile\_change}$	9F 80 12	Hilfsmittelmanager	↔
$T_{application\_info\_enq}$	9F 80 20	Anwendungsinformation	→
$T_{application\_info}$	9F 80 21	Anwendungsinformation	←
T_{enter_menu}	9F 80 22	Anwendungsinformation	→
$T_{ca\_info\_enq}$	9F 80 30	CA-Unterstützung	→
T_{ca_info}	9F 80 31	CA-Unterstützung	←
T_{ca_pmt}	9F 80 32	CA-Unterstützung	→
$T_{ca\_pmt\_reply}$	9F 80 33	CA-Unterstützung	←
T_{tune}	9F 84 00	Hauptgerätesteuerung	←
$T_{replace}$	9F 84 01	Hauptgerätesteuerung	←
$T_{clear\_replace}$	9F 84 02	Hauptgerätesteuerung	←
$T_{ask\_release}$	9F 84 03	Hauptgerätesteuerung	→
$T_{date\_time\_enq}$	9F 84 40	Datum-Uhrzeit	←
T_{date_time}	9F 84 41	Datum-Uhrzeit	→
T_{close_mmi}	9F 88 00	Mensch-Maschine-Schnittstelle	→
$T_{display\_control}$	9F 88 01	MMI	←
$T_{display\_reply}$	9F 88 02	MMI	→
T_{text_last}	9F 88 03	MMI	←
T_{text_more}	9F 88 04	MMI	←
$T_{keypad\_control}$	9F 88 05	MMI	←
$T_{keypress}$	9F 88 06	MMI	→
T_{enq}	9F 88 07	MMI	←
T_{answ}	9F 88 08	MMI	→
T_{list_last}	9F 88 0C	MMI	←
T_{list_more}	9F 88 0D	MMI	←
$T_{subtitle\_segment\_last}$	9F 88 0E	MMI	←
$T_{subtitle\_segment\_more}$	9F 88 0F	MMI	→
$T_{display\_message}$	9F 88 10	MMI	←
$T_{scene\_end\_mark}$	9F 88 11	MMI	←
T_{scene_done}	9F 88 12	MMI	←
$T_{scene\_control}$	9F 88 13	MMI	→
$T_{subtitle\_download\_last}$	9F 88 14	MMI	→
$T_{subtitle\_download\_more}$	9F 88 15	MMI	←
$T_{flush\_download}$	9F 88 16	MMI	←
$T_{download\_reply}$	9F 88 17	MMI	←
T_{comms_cmd}	9F 8C 00	Kommunikation niedriger Datenrate	←
$T_{connection\_descriptor}$	9F 8C 01	Kommunikation niedriger Datenrate	←
T_{comms_reply}	9F 8C 02	Kommunikation niedriger Datenrate	→
$T_{comms\_send\_last}$	9F 8C 03	Kommunikation niedriger Datenrate	←
$T_{comms\_send\_more}$	9F 8C 04	Kommunikation niedriger Datenrate	←
$T_{comms\_rcv\_last}$	9F 8C 05	Kommunikation niedriger Datenrate	→
$T_{comms\ rcv\_more}$	9F 8C 06	Kommunikation niedriger Datenrate	→

Anhang A
(normativ)

Auf der PC-Karte basierende physikalische Schicht

A.1 Allgemeine Beschreibung

Die physikalische Schnittstelle zwischen Modul und Hauptgerät, der Formfaktor und mechanische Eigenschaften des Moduls bilden eine Variante der PC-Kartenspezifikation [6], [7] und [8]. Bild A.1 zeigt eine typische Modularchitektur.

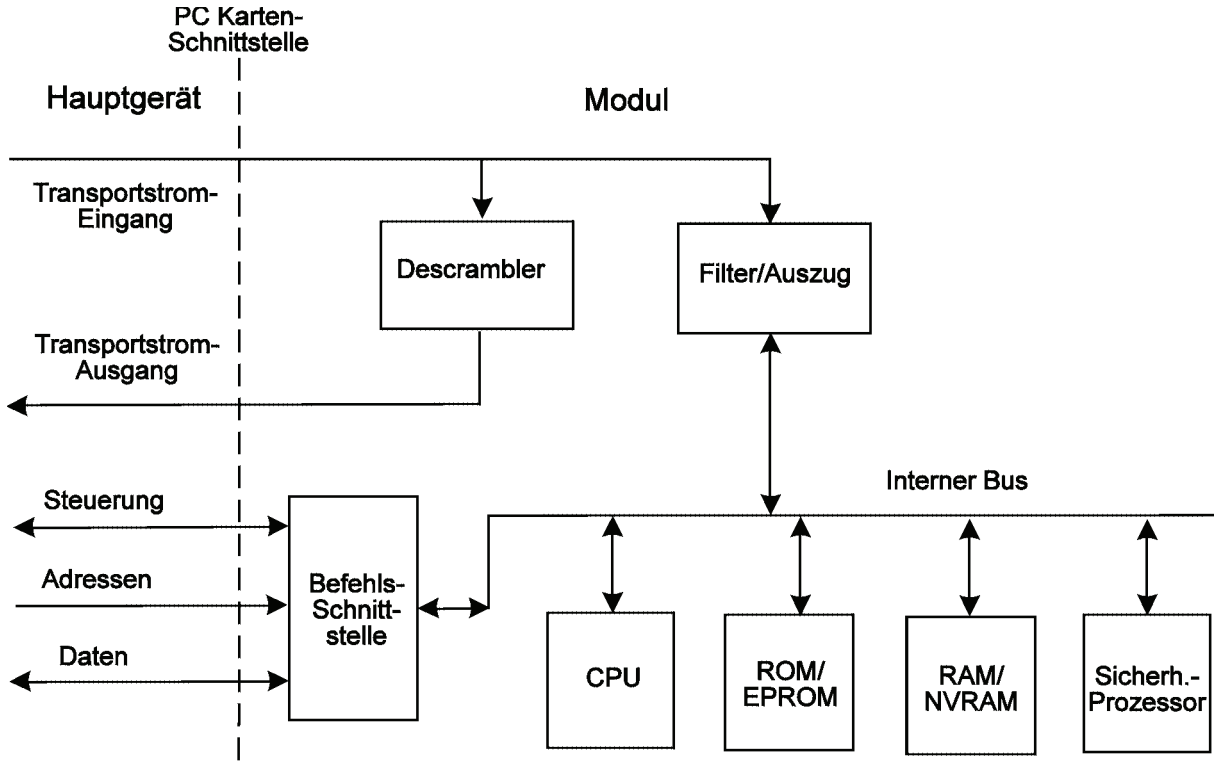

Bild A.1 – Typische CA-Modularchitektur mit der Lage der PC-Kartenschnittstelle

A.1.1 PC-Kartenschnittstelle

Die PC-Karten-Schnittstelle wird in dem folgenden Abschnitt beschrieben. Sie ist eine Variante der PC-Karten-Spezifikation.

Die Variante bietet die folgenden Leistungsmerkmale:

– Transportstromschnittstelle für MPEG-2-Daten, bestehend aus einem parallelen 8-Bit-Eingang zu dem Modul und einem getrennten parallelen 8-Bit-Ausgang, zusammen mit Steuersignalen und einer Byte-Synchronisierung.

– Befehlsschnittstelle für Befehlsverkehr zwischen Hauptgerät und Modul, bestehend aus einem bidirektionalen 8-Bit-Datenbus mit Adressen- und Steuersignalen.

– Attributspeicher-Schnittstelle, um dem Hauptgerät zu ermöglichen, die Karten-Informationsstruktur in dem Modul zu lesen und das Modul in seiner üblichen Betriebsart zu konfigurieren.

A.1.2 Descrambler

Der Descrambler descrambelt selektiv Transportpakete innerhalb des Transportstromes. Wenn er dazu konfiguriert ist, kann er auch auf der Ebene des paketierten Elementarstroms descrambeln. Er wird durch spezielle Werte (Steuerworte) konfiguriert, die von der CPU periodisch in ihn geladen werden. Da der Descrambler mehrere Dienste gleichzeitig descrambeln können muss, führt er üblicherweise eine Liste der Steuerworte mit den zugeordneten Paketkennungen (PID), die descrambelt werden müssen. Der Descrambler descrambelt keine Transportpakete, bei denen das transport_scrambling_control-flag (Transportstrom-Scrambling-Steuerungs-Flag) auf '00' gesetzt ist, auch wenn die PID zu dem aktuellen Steuerwort passt. Dieselbe Einschränkung gilt im Hinblick auf das PES_scrambling_control-flag (Paketierter-Elementarstrom-Scrambling-Flag) für Scrambling auf der PES-Ebene.

A.1.3 Filter/Auszug

Einige der Daten, die für erfolgreichen Betrieb des CA-Systems erforderlich sind, werden innerhalb des Transportstromes übertragen. Die Filter/Auszug-Schaltung wird so konfiguriert, dass aus dem Transportstrom die Daten gewonnen werden, die von dem CA-Modul für die zu descrambelnden Programme/Dienste benötigt werden.

A.1.4 CPU

Sie lässt den CA-Prozess ablaufen und steuert den Datenfluss innerhalb des Moduls und zwischen Modul und Hauptgerät zur Übertragung der CA-Funktionen.

A.1.5 ROM/EPROM und RAM/NVRAM

Sie enthalten das Programm und Daten, die den CA-Prozess implementieren. Das ROM kann auch den Attributespeicher enthalten, der für den Initialisierungsprozess der PC-Karte benötigt wird.

A.1.6 Sicherheitsprozessor

Dies ist ein separater Prozessor, der mit wesentlich höheren Sicherheitsanforderungen als die Haupt-CPU hergestellt wird. Er überträgt Sicherheitsfunktionen, wie die Entschlüsselung und speichert auch Sicherheitsinformationen, wie Schlüssel und Berechtigungen. Er kann in das PC-Kartenmodul eingebettet sein oder er kann sich in einem dazugehörenden abnehmbaren Modul wie einer Chipkarte befinden.

A.2 Elektrische Schnittstelle

Die elektrischen Kennwerte dieser Schnittstelle entsprechen der angegebenen Variante der Version 2.1 der CP-Kartennorm (siehe A.5).

A.2.1 Transportstrom-Schnittstelle

MPEG-2-Daten vom Hauptgerät werden dem Modul auf einem 8-Bit-Datenbus MDI0 bis MDI7 übergeben. Zusätzlich gibt es zwei Steuersignale MISTRT und MIVAL. Diese werden durch ein Taktsignal MCLK in das Modul getaktet. Von dem Modul kommen die MPEG-2-Daten auf einem weiteren 8-Bit-Datenbus MDO0 bis MDO7 zurück. Entsprechend gibt es zwei Steuersignale MOSTRT und MOVAL. Eine genaue Beschreibung dieser Schnittstelle wird in A.5 gegeben.

A.2.2 Befehlsschnittstelle

A.2.2.1 Hardware-Schnittstellenbeschreibung

Die Hardwareschnittstelle besteht aus mehreren Registern, die 4 Bytes des I/O-Adressenraumes auf der PC-Karten-Schnittstelle belegen. Byteoffset 0 ist das Datenregister. Dies wird gelesen, um Daten vom Modul zu übermitteln, und geschrieben, um Daten an das Modul zu übermitteln. Byteoffset 1 ist das Steuerregister und das Statusregister. Lesen bei Offset 1 liest das Statusregister und Schreiben bei Offset 1 schreibt in das Steuerregister. Das Größenregister ist ein 16-Bit-Register bei Byteoffset 2 und 3. Offset 2 ist die niedrigstwertige Hälfte und Offset 3 die höchstwertige Hälfte. Eine Abbildung der Register wird in Bild A.2 gezeigt.

Nur zwei Adressleitungen, A0 und A1 werden von der Schnittstelle decodiert. Dem Entwickler des Hauptgerätes ist es freigestellt diesen Block von 4 Bytes irgendwo innerhalb des eigenen Adressenbereiches durch geeignete Decodierung oder Konvertierung von anderen Adressleitungen innerhalb des Hauptgerätes unterzubringen.

Offset	Register
0	Datenregister
1	Befehls/Status-Register
2	Größenregister (LS)
3	Größenregister (MS)

Bild A.2 – Abbildung der Register der Hardwareschnittstelle

Das Statusregister sieht wie folgt aus:

Bit	7	6	5	4	3	2	1	0
	DA	FR	R	R	R	R	WE	RE

DA (Daten verfügbar) wird auf '1' gesetzt, wenn das Modul bestimmte Daten an das Hauptgerät zu senden hat.

FR (frei) wird auf '1' gesetzt, wenn das Modul frei ist, um Daten vom Hauptgerät anzunehmen, und beim Abschluss des Rücksetzvorgangs, der entweder durch ein Modul-Hardware-Rücksetzen oder durch den RS-Befehl ausgelöst wurde.

R sind Reservebits. Sie werden als Null gelesen.

WE (Schreibfehler) und RE (Lesefehler) werden benutzt, um Längenfehler von Lese- oder Schreiboperationen anzuzeigen.

Die DA- und FR-Signale können auch, durch Interrupt-Freigabebits im Befehlsregister gesteuert, Interrupts erzeugen.

Das Befehlsregister sieht wie folgt aus:

Bit	7	6	5	4	3	2	1	0
	DAIE	FRIE	R	R	RS	SR	SW	HC

DAIE wird auf '1' gesetzt, um das DA-Signal freizugeben, das IREQ#-Signal an der PC-Karten-Schnittstelle zu aktivieren.

FRIE wird auf '1' gesetzt, um das FR-Signal freizugeben, das IREQ#-Signal an der PC-Karten-Schnittstelle zu aktivieren.

RS (Rücksetzen) wird auf '1' gesetzt, um die Schnittstelle zurückzusetzen. RS setzt nicht das ganze Modul zurück.

SR (Größe Lesen) wird auf '1' gesetzt, um das Modul aufzufordern, seine maximale Puffergröße bereitzustellen. Wird vom Hauptgerät nach der Datenübertragung auf '0' gesetzt.

SW (Größe Schreiben) wird auf '1' gesetzt, um dem Modul mitzuteilen, welche Puffergröße es verwenden muss. Wird vom Hauptgerät nach der Datenübertragung auf '0' gesetzt.

HC (Hauptgerätesteuerung) wird durch das Hauptgerät auf '1' gesetzt, bevor eine Daten-Schreibsequenz gestartet wird. Wird vom Hauptgerät nach der Datenübertragung auf '0' gesetzt.

R sind Reservebits. Sie werden immer als Null geschrieben.

A.2.2.1.1 Initialisierung

Während der Initialisierung des Moduls und wenn sonst ein Fehler aufgetreten ist, muss das Hauptgerät in der Lage sein, die Schnittstelle zurückzusetzen. Es tut dies durch Schreiben einer '1' auf das RS-Bit im Steuerregister und durch nochmaliges Schreiben einer '0' auf das RS-Bit. Es muss ein Impuls von mindestens $40\ \mu s$ Dauer gewährleistet sein. Das Modul löscht alle Daten in seinen Datenübertragungspuffern und stellt die Schnittstelle so ein, dass sie das Puffergrößen-Verhandlungsprotokoll durchführt. Das Modul signalisiert durch Setzen des FR-Bits auf '1', dass die Rücksetzoperation abgeschlossen ist.

Nach der Initialisierung muss das Hauptgerät die Größe des internen Puffers des Moduls mit Hilfe des Puffergrößen-Verhandlungsprotokolls herausfinden. Weder Hauptgerät noch Modul dürfen die Schnittstelle zur Datenübertragung benutzen, bis dieses Protokoll abgeschlossen ist. Das Hauptgerät beginnt die Verhandlung durch Schreiben einer '1' auf das SR-Bit in dem Steuerregister, das auf das zu setzende DA-Bit wartet und liest dann die Puffergröße, wie nachstehend beschrieben, durch einen Übertragungsvorgang Modul zu Hauptgerät. Am Ende des Übertragungsvorganges setzt das Hauptgerät das SR-Bit auf '0' zurück. Die Antwortdaten sind 2 Bytes mit dem MSB zuerst. Module müssen eine Mindestpuffergröße von 16 Bytes unterstützen. Der Höchstwert ist durch die Begrenzung der Registergröße (65535 Bytes) festgelegt. Ähnlich kann das Hauptgerät eine Begrenzung der von ihm vorgegebenen Puffergröße haben. Das Hauptgerät muss eine Mindestpuffergröße von 256 Bytes unterstützen, es können aber bis zu 65535 Bytes sein. Nach Lesen der Puffergröße, die das Modul unterstützen kann, nimmt das Hauptgerät den kleineren Wert seiner eigenen Puffergröße und der Modul-Puffergröße. Dies wird die Puffergröße für alle nachfolgenden Datenübertragungen zwischen Hauptgerät und Modul. Das Hauptgerät teilt nun dem Modul mit, diese Puffergröße durch Schreiben einer '1' auf das SW-Bit im Befehlsregister zu benutzen, und wartet, bis das FR-Bit gesetzt ist, und schreibt dann die Größe als 2-Datenbytes, das höchstwertige Byte zuerst, und benutzt die nachstehend beschriebenen Übertragungsvorgänge Hauptgerät zu Modul. Am Ende der Übertragung setzt das Hauptgerät das SW-Bit auf '0'. Die ausgehandelte Puffergröße gilt für beide Richtungen des Datenflusses, sogar bei Implementierungen mit zwei Puffern.

A.2.2.1.2 Übertragung Hauptgerät zu Modul

Das Hauptgerät setzt das WR-Bit und prüft dann das FR-Bit. Wenn FR '0' ist, dann ist die Schnittstelle belegt und das Hauptgerät muss HC zurücksetzen und eine Zeitspanne warten, bevor es die Prüfung wiederholt. Wenn FR '1' ist, dann schreibt das Hauptgerät die Anzahl der Bytes, die zum Senden an das Modul vorliegen,

in das Größenregister und schreibt dann die Anzahl der Datenbytes in das Datenregister. Dieses mehrfache Schreiben darf durch keinen anderen Betrieb an der Schnittstelle unterbrochen werden, außer zum Lesen des Statusregisters. Wenn das erste Byte geschrieben ist, setzt das Modul WE auf '1' und FR auf '0'. Während der Übermittelung bleibt WE auf '1', bis das letzte Byte geschrieben und an dieser Stelle WE auf '0' gesetzt wird. Wenn weitere Bytes geschrieben werden, dann wird das WE-Bit auf '1' gesetzt. Am Ende der Übertragung muss das Hauptgerät das HC-Bit durch Schreiben von '0' zurücksetzen.

Das Hauptgerät muss das DA-Bit prüfen, bevor es den oben beschriebenen Hauptgerät-zu-Modul-Zyklus einleitet, um im Fall der Implementierung eines einzelnen Puffers im Modul eine Blockierung zu vermeiden.

Dieser Programmteil in C veranschaulicht den Prozess auf der Seite des Hauptgerätes:

```
if (Status_Reg & 0x80) /*gehe zu Modul-zu-Hauptgeräte-Übertragung */
Command_Reg = 0x01;
if (Status _Reg & 0x40) {
   Size_Reg [0] = bsize & 0xFF;
   Size_Reg [1] = bsize >> 8;
   for (i=0; i<bsize; i++)
      Data_Reg = write_buf [i];
}
Command_Reg = 0x00;
```

A.2.2.1.3 Übertragung Modul zu Hauptgerät

Das Hauptgerät wartet auf ein DA-Interrupt oder prüft das DA-Bit im Statusregister. Wenn DA auf '1' gesetzt ist, dann liest das Hauptgerät das Größenregister und ermittelt, wie viel Daten zu übermitteln sind. Dann liest es die Anzahl der Bytes vom Datenregister. Dieses mehrfache Lesen darf durch keinen anderen Betrieb an der Schnittstelle unterbrochen werden, außer zum Lesen des Statusregisters. Wenn das erste Byte gelesen ist, setzt das Modul RE auf '1' und DA auf '0'. Während der Übertragung bleibt RE auf '1', bis das letzte Byte geschrieben und RE an dieser Stelle auf '0' gesetzt wird. Wenn weitere Bytes geschrieben werden, dann wird das RE-Bit auf '1' gesetzt.

Das C-Programm veranschaulicht den Vorgang auf der Seite des Hauptgerätes:

```
if (Status_Reg & 0x80) {
   bsize = Size_Reg [0]  |  Size_Reg[1] << 8;
   for  (i=0; i<bsize; i++)
   read_buf [i] = Data_Reg;
}
```

Die Bytes des Größenregisters können in jeder Reihenfolge gelesen oder geschrieben werden.

A.2.2.1.4 Interrupts

Unterstützung der Interrupts durch Module ist obligatorisch. Hauptgeräte können interrupt-gesteuerte I/O oder nach Wunsch zyklisch abgefragte I/O verwenden. Der Interruptbetrieb findet wie folgt statt:

Wenn entweder DA oder FR aktiviert wird und das passende Interrupt-Freigabebit gesetzt ist, dann wird IREQ# aktiviert. Eine Interruptroutine wird aufgerufen, die dann beide Statusbits prüft und die passende Aktion vornimmt. Das Hauptgerät muss darauf vorbereitet sein, das FR-Bit rückgesetzt vorzufinden, wenn es geprüft wird, obwohl ein Interrupt erkannt wurde, wie bei einer Lösung mit einem einzigen Puffer kann das Modul den freien Puffer belegt haben, bevor der Interrupt bedient wurde.

A.2.2.2 Hardwareunterstützung im Modul

Jede Implementierung, die die obige Spezifikation für den Hauptgerät-Modul-Dialog berücksichtigt, ist möglich. Das Modul könnte einen einzigen Puffer verwenden, der für Transaktionen in beiden Richtungen benutzt wird, oder, um die Schnittstelle zu beschleunigen, kann für jede Flussrichtung ein Puffer verwendet werden. Durch entsprechende Steuerung der Wechselwirkung zwischen dem Füllzustand der Puffer, dem WR-Bit und dem FR- und DA-Bit können beide Schnittstellenkonzeptionen angepasst werden.

A.3 Verbindungsschicht

A.3.1 Transportstromschnittstelle

An der Transportstromschnittstelle gibt es keine Verbindungsschicht. Die Daten liegen in Form von aufeinander folgenden MPEG-2-Transportpaketen vor, möglicherweise mit Datenlücken innerhalb und zwischen den Transportpaketen.

A.3.2 Befehlsschnittstelle

Die Verbindungsschicht an der Befehlsschnittstelle erfüllt zwei Aufgaben. Sie fragmentiert, wenn erforderlich, Transportprotokoll-Dateneinheiten (TPDU) zum Senden über die Puffer begrenzter Größe der physikalischen Schicht, und setzt die empfangenen Teilstücke wieder zusammen. Sie multiplext auch verschiedene Transportverbindungen einwandfrei in die eine Leitungsverbindung. Sie führt dies durch Verschachteln der Teilstücke von allen Transportverbindungen durch, die im Augenblick zum Senden von TPDUs über die Leitung bereitliegen. Die Verbindungsschicht setzt voraus, dass der Übertragungsmechanismus der physikalischen Schicht zuverlässig ist, d. h. dass er die Daten in der richtigen Reihenfolge hält und weder welche löscht noch welche wiederholt.

Eine Leitungsverbindung wird als Konsequenz der Errichtung der Verbindung der physikalischen Schicht automatisch eingerichtet, d. h. Einstecken des Moduls oder Einschalten der Stromversorgung, Lesen der Karten-Informationsstruktur und Konfigurieren des Moduls in der geeigneten Betriebsart. Es ist kein weiteres spezielles Einrichtungsverfahren erforderlich. Die Größe jeder Leitungsprotokoll-Dateneinheit (LPDU) hängt von der Größe ab, die das Hauptgerät und das Modul unter Benutzung der SR- und SW-Befehle an der Schnittstelle aushandeln. Jede LPDU besteht aus einem 2-Byte-Header, gefolgt von einem Teilstück einer TPDU, die Gesamtgröße übersteigt nicht die ausgehandelte Puffergröße. Das erste Byte des Headers ist die Transportverbindungskennung für dieses TPDU-Teilstück. Das zweite Byte enthält einen 'Mehr/Letzter'-Indikator in seinem höchstwertigen Bit. Wenn das Bit auf '1' gesetzt ist, dann folgt mindestens noch ein TPDU-Teilstück, und wenn das Bit auf '0' gesetzt ist, dann zeigt es, das dies das letzte (oder einzige) Teilstück der TPDU für diese Transportverbindung ist. Alle anderen Bits in dem zweiten Byte sind reserviert und müssen auf Null gesetzt sein. Dies veranschaulicht das Bild A.3.

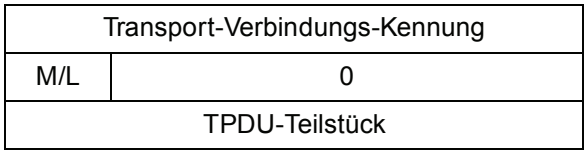

Bild A.3 – Anordnung der Leitungsprotokoll-Dateneinheit

Jede TPDU muss in einer neuen LPDU beginnen, d. h. die LPDU, die das letzte Teilstück der vorhergehenden TPDU auf einer Transportverbindung überträgt, kann nicht das erste Teilstück der nächsten übertragen. Wenn mehr als eine Transportverbindung im Augenblick TPDUs zu transportieren hat, muss die Verbindungsschicht für jede davon der Reihe nach ein Teilstück senden, so dass alle Transportverbindungen eine gerechte Zuteilung der verfügbaren Kommunikationsbandbreite erhalten.

A.4 Implementierungsspezifische Transport-Unterschicht über PC-Karten-Schnittstelle

A.4.1 Transportprotokoll-Objekte

Das Transportprotokoll an der Befehlsschnittstelle ist ein Befehl-Antwort-Protokoll, bei dem das Hauptgerät unter Benutzung einer Befehls-Transportprotokoll-Dateneinheit (C_TPDU) einen Befehl an das Modul sendet und auf die Antwort mit einer Antwort-Transportprotokoll-Dateneinheit (R_TPDU) von dem Modul wartet. Das Modul kann keine Kommunikation einleiten: es muss auf das Hauptgerät warten, das es abfragt oder seine Daten zuerst sendet. Das Protokoll wird von elf Transportschichtobjekten unterstützt. Einige davon treten nur in C_TPDUs vom Hauptgerät auf, einige nur in R_TPDUs von dem Modul und einige können in beiden auftreten. Create_T_C und C_T_C_Reply erstellen neue Transportverbindungen. Delete_T_C und D_T_C_Reply löschen sie. Request_T_C und New_T_C ermöglichen dem Modul, beim Hauptgerät das Erstellen einer neuen Transportverbindung anzufordern. T_C_Error erlaubt Fehlerbedingungen zu signalisieren. T_SB überträgt Statusinformation vom Modul zum Hauptgerät. T_RCV fordert auf Abruf liegende Daten von einem Modul und T_Data_More und T_Data_Last transportieren Daten von höheren Schichten zwischen Hauptgerät und Modul. T_Data_Last mit einem leeren Datenfeld wird vom Hauptgerät benutzt, um regelmäßig Daten vom Modul abzufragen, wenn es selbst nichts zu senden hat. In allen Objekten gibt es ein nach den in [3] definierten Regeln codiertes tag_field und length_field, und ein t_c_id-Feld als einzelnes Oktet. Die Codierung des length_field wird in Bild 6 gezeigt.

A.4.1.1 Befehls-Transportprotokoll-Dateneinheit (C_TPDU)

Diese Befehls-TPDU (C_TPDU) transportiert Transportprotokoll-Objekte vom Hauptgerät zum Modul.

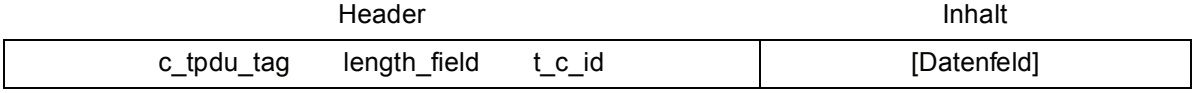

Bild A.4 – Struktur der C_TPDU

Tabelle A.1 – Codierung der C_TPDU

Syntax	Anzahl der Bits	Mnemonik
C_TPDU() {		
c_tpdu_tag	**8**	**uimsbf**
length_field()		
t_c_id	**8**	**uimsbf**
for (i=0;i<length_value-1;i++) {		
data_byte	**8**	**uimsbf**
}		
}		

Die C_TPDU besteht aus zwei Teilen:

– einem obligatorischen Header bestehend aus einem Tag-Wert c_tpdu_tag, der den TPDU-Befehl codiert, einem length_field, das die Länge aller folgenden Felder codiert, und einer Transportverbindungskennung t_c_id;

– einem bedingten Hauptteil variabler Länge, gleich der Länge, die durch length_field minus eins codiert wird.

A.4.1.2 Antwort-Transportprotokoll-Dateneinheit (R_TPDU)

Die Antwort-TPDU (R_TPDU) übermittelt Transportprotokoll-Objekte vom Modul zum Hauptgerät.

Header			Hauptteil	Status		
r_tpdu_tag	length_field	t_c_id	[Datenfeld]	SB_tag	length_field=2	SB_value

Bild A.5 – Struktur der R_TPDU

Tabelle A.2 – Codierung der R_TPDU

Syntax	Anzahl der Bits	Mnemonik
R_TPDU() {		
r_tpdu_tag	**8**	**uimsbf**
length_field()		
t_c_id	**8**	**uimsbf**
for (i=0;i<length_value-1;i++) {		
data_byte	**8**	**uimsbf**
}		
SB_tag	**8**	**uimsbf**
length_field()=2	**8**	**uimsbf**
t_c_id	**8**	**uimsbf**
SB_value	**8**	**uimsbf**
}		

Die R_TPDU besteht aus drei Teilen:

– einem bedingten Header, bestehend aus einem Tag-Wert r_tpdu_tag, der die TPDU_Antwort codiert, einem Längenfeld, das die Länge der folgenden Transportverbindungskennung und der Datenfelder codiert, und einer Transportverbindungskennung t_c_id. Der Status ist in der Berechnung des length_field nicht enthalten;

– einem bedingten Hauptteil variabler Länge gleich der Länge, die durch length_field minus eins codiert wird;

– einem obligatorischen Status, bestehend aus einem Status-Tag 'SB_tag', einem length_field gleich 2, einer Transportverbindungs-Kennung und einem Ein-Byte-Statusbytewert (SB_value), der nach folgendem Bild A.6 und Tabelle A.3 codiert ist.

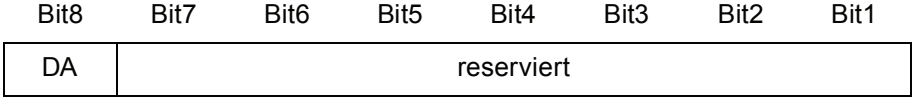

Bild A.6 – Codierung von SB_value

Das 1-Bit-DA(Daten vorhanden)-Indikatorfeld zeigt an, ob in dem Ausgangspuffer des Moduls eine Nachricht für das Hauptgerät vorhanden ist. Das Hauptgerät muss eine Receive_data C_TPDU ausgeben, um die Meldung zu übernehmen (siehe A.4.1.11.2). Die Codierung der Daten des DA-Indikators wird in Tabelle A.3 angegeben. Das 'Reserve'-Feld muss auf Null gesetzt werden.

Tabelle A.3 – Codierung Bit8 von SB_value

Bit8	Bedeutung
0 1	keine Meldung vorhanden Meldung vorhanden

A.4.1.3 Verkettung von Befehls- oder Antwort-TPDU-Datenfeldern

Das Datenfeld, das sich in einer C_TPDU oder einer R_TPDU befindet, darf, wenn es wegen der Größe der Übertragungs- oder Empfangspuffer des Hauptgerätes oder des Moduls erforderlich ist, in mehrere Blöcke geringerer Größe aufgeteilt werden.

Die Verkettung wird bei C_TPDU durch Benutzen von zwei verschiedenen c_TPDU_tag-Werten (M_c_TPDU_tag und L_c_TPDU_tag) und bei R_TPDU durch Benutzen von zwei verschiedenen r_TPDU_tag-Werten (M_r_TPDU_tag und L_r_TPDU_tag) ausgeführt. Alle Blöcke des Datenfeldes, mit Ausnahme des letzten, werden innerhalb einer C_TPDU (oder R_TPDU) mit einem M_c_TPDU-tag- (oder M_r_TPDU_tag)-Wert gesendet. Dieser Tag-Wert zeigt an, dass weitere Daten in einer weiteren C_TPDU (oder R_TPDU) gesendet werden. Der letzte Block des Datenfeldes wird innerhalb einer C_TPDU (oder R_TPDU) mit einem L_c_TPDU_tag-(oder L_r_TPDU_tag-)Wert gesendet. Wenn der letzte Block empfangen ist, verkettet die Empfangseinrichtung alle empfangenen Datenfelder.

Dieser Mechanismus ist für alle C_TPDU (oder R_TPDU) mit zwei definierten Tag-Werten gültig. Er ist in nachstehenden Bildern A.7 und A.8 veranschaulicht.

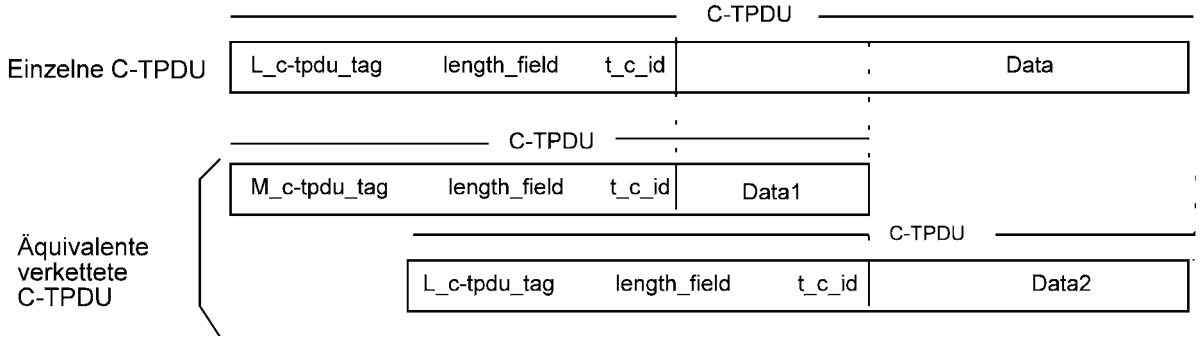

Bild A.7 – Darstellung des Verkettungsmechanismus mit C_TPDU

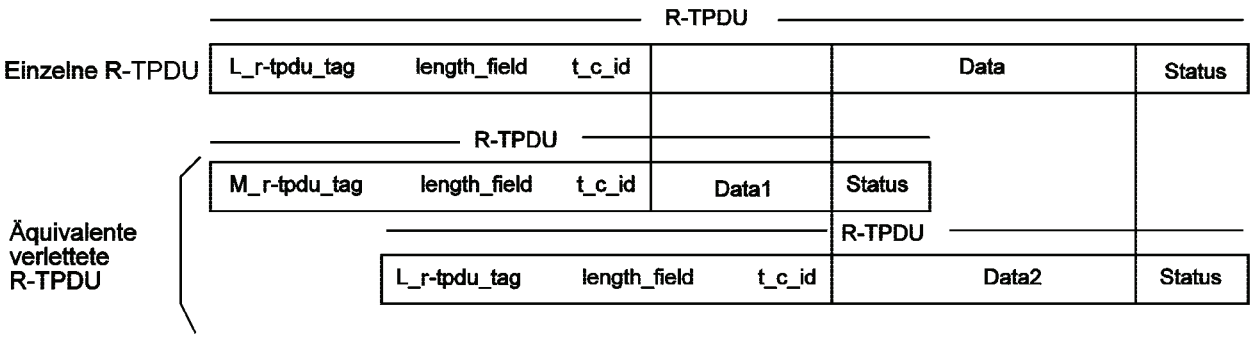

Bild A.8 – Darstellung des Verkettungsmechanismus mit R_TPDU

A.4.1.4 Transportverbindung Erstellen (Create_T_C)

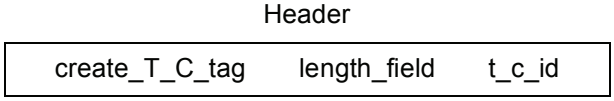

Bild A.9 – Struktur von Create_T_C

Tabelle A.4 – Codierung von Create_T_C

Syntax	Anzahl der Bits	Mnemonik
Create_T_C() {		
create_T_C_tag	**8**	**uimsbf**
length_field()		
t_c_id	**8**	**uimsbf**
}		

Das Objekt Create_T_C besteht nur aus einem Teil:

– einem obligatorischen Header, bestehend aus einem Tag-Wert create_T_C_tag, der das Objekt Create_T_C codiert, einem length_field gleich 1 und einer Transportverbindungskennung t_c_id.

A.4.1.5 Transportverbindung Erstellen-Antwort (C_T_C_Reply)

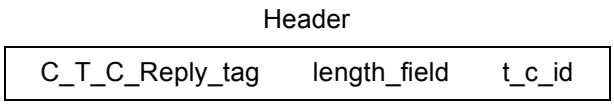

Bild A.10 – Struktur von C_T_C_Reply

Tabelle A.5 – Codierung von C_T_C_Reply

Syntax	Anzahl der Bits	Mnemonik
C_T_C_Reply() {		
C_T_C_Reply_tag	**8**	**uimsbf**
length_field()		
t_c_id	**8**	**uimsbf**
}		

Das Objekt C_T_C_Reply besteht nur aus einem Teil:

– einem obligatorischen Header, bestehend aus einem Tag-Wert C_T_C_Reply_tag, der das Objekt C_T_C_Reply codiert, einem length_field gleich 1 und einer Transportverbindungskennung.

A.4.1.6 Transportverbindung Löschen (Delete_T_C)

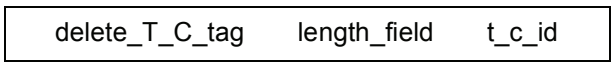

Bild A.11 – Struktur von Delete_T_C

Tabelle A.6 – Codierung von Delete_T_C

Syntax	Anzahl der Bits	Mnemonik
Delete_T_C() {		
delete_T_C_tag	**8**	**uimsbf**
length_field()		
t_c_id	**8**	**uimsbf**
}		

Das Objekt Delete_T_C besteht aus nur einem Teil:

– einem obligatorischen Header, bestehend aus einem Tag-Wert delete_T_C_tag, der das Objekt Delete_T_C codiert, einem length_field gleich 1 und einer Transportverbindungskennung.

A.4.1.7 Transportverbindung Löschen-Antwort (D_T_C_Reply)

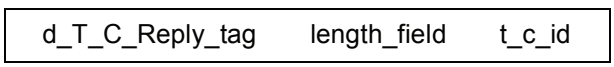

Bild A.12 – Struktur von D_T_C_Reply

Tabelle A.7 – Codierung von D_T_C_Reply

Syntax	Anzahl der Bits	Mnemonik
D_T_C_Reply() {		
d_T_C_Reply_tag	**8**	**uimshf**
length_field() = 1		
t_c_id	**8**	**uimsbf**
}		

Das Objekt D_T_C_Reply besteht nur aus einem Teil:

– einem obligatorischen Header, bestehend aus einem Tag-Wert d_T_C_Reply_tag, der das Objekt D_T_C_Reply codiert, einem length_field gleich 1 und einer Transportverbindungskennung.

A.4.1.8 Transportverbindung Anfordern (Request_T_C)

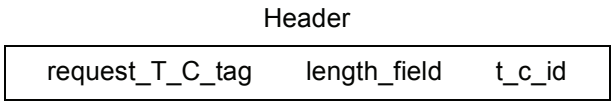

Bild A.13 – Struktur von Request_T_C

Tabelle A.8 – Codierung von Request_T_C

Syntax	Anzahl der Bits	Mnemonik
Request_T_C() {		
request_T_C_tag	**8**	**uimsbf**
length_field() = 1		
t_c_id	**8**	**uimsbf**
}		

Das Objekt Request_T_C besteht nur aus einem Teil:

– einem obligatorischen Header, bestehend aus einem Tag-Wert request_T_C_tag, der das Objekt Request_T_C codiert, einem length_field gleich 1 und einer Transportverbindungskennung.

A.4.1.9 Neue Transportverbindung (New_T_C)

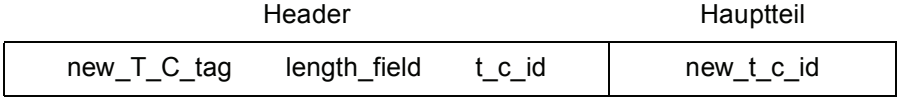

Bild A.14 – Struktur von New_T_C

Tabelle A.9 – Codierung von New_T_C

Syntax	Anzahl der Bits	Mnemonik
New_T_C() {		
new_T_C_tag	**8**	**uimsbf**
length_field() = 2		
t_c_id	**8**	**uimsbf**
new_t_c_id	**8**	**uimsbf**
}		

Das Objekt New_T_C besteht aus zwei Teilen:

– einem obligatorischen Header, bestehend aus einem Tag-Wert new_T_C_tag, der das Objekt New_T_C codiert, einem length_field gleich 2 und einer Transportverbindungskennung;

– einem obligatorischen Inhalt, bestehend aus der Transportverbindungskennung für die aufzubauende neue Verbindung.

A.4.1.10 Transportverbindungsfehler (T_C_Error)

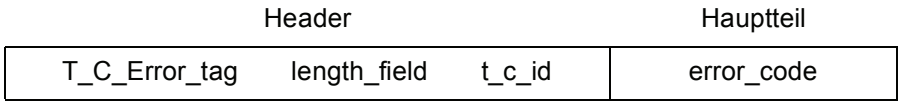

Bild A.15 – Struktur von T_C_Error

Tabelle A.10 – Codierung von T_C_Error

Syntax	Anzahl der Bits	Mnemonik
T_C_Error() {		
T_C_Error_tag	**8**	**uimsbf**
length_field() = 2		
t_c_id	**8**	**uimsbf**
error-code	**8**	**uimsbf**
}		

Das Objekt T_C_Error besteht aus zwei Teilen:

– einem obligatorischen Header, bestehend aus einem Tag-Wert T_C_Error_tag, der das Objekt T_C_Error codiert, einem length_field gleich 2 und einer Transportverbindungskennung;

– einem obligatorischen Inhalt, bestehend aus dem Fehlercode für den zu signalisierenden speziellen Fehler.

Die benutzten Fehlercodes sind folgende:

Tabelle A.11 – Fehlercodewerte

Fehlercode	Bedeutung
1	Keine Transportverbindungen verfügbar

A.4.1.11 Liste der C_TPDUs und der dazugehörenden R_TPDUs

A.4.1.11.1 Befehl Daten Senden

Dieser Befehl wird vom Hauptgerät benutzt, wenn die Transportverbindung offen ist, entweder um Daten zum Modul zu senden oder durch das Statusbyte gegebene Information zu holen. Das Modul antwortet mit dem Statusbyte.

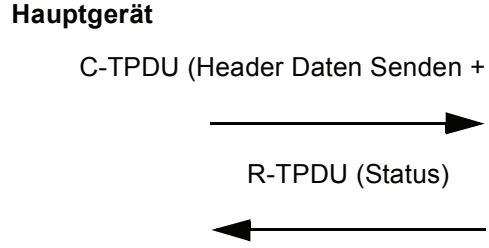

Bild A.16 – Befehl/Antwort-Paar Daten Senden

Tabelle A.12 – Codierung der Daten Senden-C_TPDU

c_TPDU_tag	M_c_TPDU_tag : T_{data_more} L_c_TPDU_tag : T_{data_last}
length_field	Länge des Datenfeldes entsprechend [3]
t_c_id	Transportverbindungskennung
Datenfeld	Untermenge von (TLV TLV .... TLV)

Tabelle A.13 – Codierung der Daten Senden-R_TPDU

Status	Entsprechend Bild A.6

Eine Daten-Senden-C_TPDU mit L=1 (kein Datenfeld) kann vom Hauptgerät ausgegeben werden, nur um Informationen zu holen, die von dem Statusbyte gegeben wird (siehe Abfragefunktion in A.4.1.12).

A.4.1.11.2 Befehl Daten Empfangen

Dieser Befehl wird vom Hauptgerät benutzt, um vom Modul Daten zu empfangen.

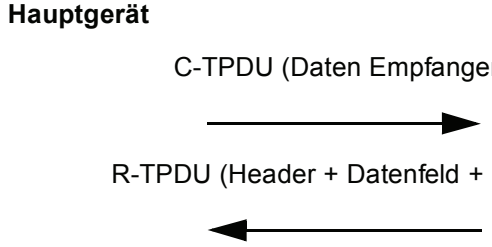

Bild A.17 – Befehls/Antwort-Paar Daten Empfangen

Tabelle A.14 – Codierung der Daten Empfangen C_TPDU

c_TPDU_tag	T_{RCV}
length_field	c_TPDU_length ist auf '1' zu setzen
t_c_id	Transportverbindungskennung

Tabelle A.15 – Codierung der Daten Empfangen R_TPDU

r_TPDU_tag	M_r_TPDU_tag : T_{data_more} L_r_TPDU_tag : T_{data_last}
length_field	Länge des Datenfeldes entsprechend [3]
t_c_id	Transportverbindungskennung
Datenfeld	Untermenge von (TLV TLV ... TLV)
Status	Entsprechend Bild A.6

A.4.1.12 Regeln für die Abfragefunktion

Die Abfragefunktion besteht aus dem Senden eines Befehls an das Modul, um von dort mitgeteilt zu bekommen, ob Daten vorliegen oder nicht, die an das Hauptgerät zu senden sind.

Diese Funktion wird vom Hauptgerät bereitgestellt, das regelmäßig eine Send data C_TPDU mit der Länge L gleich 1 (nur t_c_id-Feld, kein Datenfeld) ausgibt. Solange keine Daten vorhanden sind, antwortet das Modul mit dem Statusbyte mit auf 0 gesetztem DA-Indikator. Wenn Daten vorhanden sind, antwortet das Modul mit dem Statusbyte mit auf 1 gesetztem DA-Indikator. Das Hauptgerät kann dann eine oder mehrere Daten-Empfangen-C_TPDU ausgeben, die die Datenübertragung bewirken, bis der DA-Indikator des Statusbytes wieder auf 0 gesetzt ist. Während das Modul noch Daten zu senden hat, d. h. das Hauptgerät hat Statusbytes mit gesetztem DA-Indikator empfangen, muss die Abfragefunktion aufgehoben sein. Sie muss wieder beginnen, wenn ein Statusbyte mit nicht gesetztem DA-Indikator empfangen wird.

Die maximale Zeitspanne beim Abfragen beträgt $100\ \text{ms}$.

Bei jeder Abfrage wird eine Zeitbegrenzung von $300\ \text{ms}$ begonnen und sie wird zurückgesetzt, wenn die Abfrageantwort empfangen wird. Wenn innerhalb dieser Zeit keine Abfrageantwort empfangen wurde, dann wird die Transportverbindung vom Hauptgerät auf übliche Art gelöscht. Das Hauptgerät sendet keine zusätzlichen Abfragen, während es auf die Abfrageantwort wartet, es sei denn, die übliche Abfrage-Wiederholungszeit wird überschritten.

A.4.1.13 Liste der Transport-Tags

Die Codierung des tpdu_tag entspricht den ASN.1-Regeln. Jedes tpdu_tag wird in einem Byte codiert.

Tabelle A.16 – Codierung von Transport-Tags

tpdu_tag	Tag-Wert (hex)	Einfach (P) oder zusammen-gesetzt (C)	Richtung Hauptgerät ↔ Modul
T_{SB}	'80'	P	←
T_{RCV}	'81'	P	→
$T_{create\_t\_c}$	'82'	P	→
$T_{c\_t\_c\_reply}$	'83'	P	←
T_{delete}	'84'	P	↔
$T_{d\_t\_c\_reply}$	'85'	P	↔
$T_{request\_t\_c}$	'86'	P	←
$T_{new\_t\_c}$	'87'	P	→
$T_{t\_c\_error}$	'88'	P	→
T_{data_last}	'A0'	C	↔
T_{data_more}	'A1'	C	↔

A.5 Teilspezifikation für PC-Karten, die bei konformen Hauptgeräten und Modulen benutzt werden

A.5.1 Ziele

Ziel dieses Abschnittes ist es, alle erforderlichen Informationen zu erfassen, [6], [7], [8], die zur Interpretation der PC-Karten-Spezifikationen benötigt werden und um für die Entwicklung von minimal konformen Hauptgeräten und Modulen davon eine Auswahl zu treffen.

A.5.2 Einführung

Dieser Abschnitt definiert den Mindestanteil der PC-Karten-Spezifikation, der sowohl von Hauptgeräten als auch von Modulen verwendet wird. Nichts in diesem Abschnitt hindert Hauptgeräte am Implementieren von mehr als dem Mindestanteil, z. B. um weitere Typen von PC-Karten-Baueinheiten zu unterstützen. Alle Module müssen jedoch so implementiert werden, dass sie in Hauptgeräten einwandfrei arbeiten, die mit den hier beschriebenen Teilspezifikationen konform sind.

Dieser Abschnitt gibt umfassende Hinweise auf die Schriftstücke der PC-Karten-Spezifikation [6], [7], [8].

A.5.3 Terminologie

An verschiedenen Stellen dieses Abschnitts werden hexadezimale Zahlen angegeben. Das benutzte Format verwendet für die letzten 6 Ziffern die Großbuchstaben A bis F und wird mit einem kleinen 'h' abgeschlossen, z. B. 7FFh.

A.5.4 Physikalische Spezifikation

A.5.4.1 Abmessungen der Karten

Das Hauptgerät muss PC-Karten sowohl vom Typ I als auch vom Typ II akzeptieren (vergleiche Abschnitt 3 in [7]). Hauptgeräteunterstützung für Karten Typ III ist freigestellt.

A.5.4.2 Steckverbinder

Es gilt Abschnitt 4 in [7].

A.5.4.3 PC-Kartenführung

Es gilt Abschnitt 5 in [7].

A.5.4.4 Erdung/EMI-Klemmen

Es gilt Abschnitt 6 in [7].

A.5.4.5 Zuverlässigkeit der Steckverbinder

Es gilt Abschnitt 7 in [7].

A.5.4.6 Haltbarkeit der Steckverbinder

Es gilt 8.2 (raue Umgebung) in [7].

A.5.4.7 PC-Karten-Umgebung

Es gilt Abschnitt 9 in [7]. Mit Rücksicht auf die Temperaturspezifikation muss das Modul, wie in der PC-Karten-Spezifikation definiert, bei bis zu 55 °C betrieben werden können. Dies vor allem, um zuverlässigen Batteriebetrieb in den Modulen zu ermöglichen. Um diesen Betriebsumgebungs-Grenzwert leichter einzuhalten, muss das Hauptgerät den Temperaturanstieg zwischen der Umgebung außerhalb des Hauptgerätes und der Umgebung des Moduls auf 15 °C begrenzen, wenn das Modul seine volle Nennleistung aufnimmt. Es ist zu beachten, dass dies nicht sicherstellen kann, dass die 55-°C-Grenze für Module unter allen Umgebungsbedingungen eingehalten wird, die sonst für das Hauptgerät akzeptabel sind. Die Entwickler der Module müssen durch Beachten der Empfehlungen der PC-Kartennorm bezüglich der Anordnung der Batterien das Risiko der Fehlbedienung bei Modulen, die Batterien enthalten, so gering wie möglich halten. Die Entwickler des Hauptgerätes sollten beim Entwurf des thermischen Konzeptes des Hauptgerätes über diese Empfehlungen informiert sein. Es ist zu beachten, dass sich die Entwickler im Hinblick auf die Temperatur des Moduls nach den einschlägigen obligatorischen Sicherheitsvorschriften richten müssen.

A.5.5 Elektrische Spezifikation

Als Variante der elektrischen Schnittstelle der 16-Bit-PC-Karte wird ein Modul mit Einheitlicher Schnittstelle implementiert (Abschnitt 4 von [6]). Die Befehlsschnittstelle benutzt das geringstwertige Byte des Datenbusses zusammen mit dem unteren Teil des Adressbusses (A0 bis A14) und entsprechende Steuersignale. Die Befehlsschnittstelle arbeitet im I/O-Schnittstellenmode. Die oberen Adressenleitungen (A15 bis A25), die höchstwertige Hälfte des Datenbusses (D8 bis D15) und bestimmte andere Steuersignale werden für diese Schnittstellenvariante neu definiert.

Beim ersten Anschließen an die Stromversorgung und vor dem Konfigurieren muss ein mit dieser Schnittstellenspezifikation konformes Modul sich verhalten wie eine Nur-Speicher-Baueinheit mit den folgenden Einschränkungen:

a) Die Signale D8 bis D15 müssen hohe Impedanz behalten.

b) 16-Bit-Lese- und Schreibmodes sind nicht vorhanden. CE2# muss ignoriert und vom Modul immer als 'High' interpretiert werden.

c) Die Adressenleitungen A15 bis A25 dürfen nicht zur Verwendung als Adressenleitungen verfügbar sein. Der maximale im Modul vorhandene Adressenraum ist auf 32768 Bytes begrenzt (16384 Bytes Attributspeicher, wie er sich ergibt, wenn nur gerade Adressen verwendet werden).

d) Die Signale BVD1 und BVD2 müssen 'High' bleiben.

A.5.5.1 Erkennen des Kartentyps

Es gilt Abschnitt 3 von [6]. Hauptgeräte müssen 5-V- und dürfen wahlweise 3,3-V-Betrieb unterstützen. Im zweiten Fall müssen die Hauptgeräte den in [6] beschriebenen Erkennungsmechanismus verwenden, um die Betriebsspannung des Moduls zu ermitteln. Hauptgeräte müssen den Festlegungen für den Buchsentyp entsprechen, wenn sie für 3,3-V-Betrieb vorgesehen sind – 3.2 von [6]. Hauptgeräte brauchen den Erkennungsmechanismus für die CardBus-PC-Karten nicht zu unterstützen, dürfen es aber wahlweise – 3.3 von [6].

A.5.5.2 Stiftbelegung

Im Speicherkartenmode, direkt nach dem Rücksetzen muss die linke Spalte der Tabellen 4-1 und 4-2 von [6] benutzt werden. Wenn das Modul während des Initialisierungsprozesses als Variante der einheitlichen Schnittstelle konfiguriert wird, werden die folgenden Neuzuordnungen durchgeführt: Die Stifte, zur Übertragung der Signale A15 bis A25, D8 bis D15, BVD1, BVD2 und VS# werden benutzt, um Eingangs- und Ausgangsbusse hoher Datenrate für die MPEX-2-Multiplexdaten bereitzustellen. Alle anderen Stifte behalten ihre Belegung als Eingang/Ausgangs- und Speicherkarten-Schnittstelle, außer dass IOIS16# niemals eingesetzt und CE2# ignoriert wird.

Die Stiftbelegung für die Kundenschnittstelle wird in folgender Tabelle A.17 definiert.

Tabelle A.17 – Stiftbelegung für diese PC-Kartenvariante

Stift	Signal	I/O	Funktion
1	GND		Erde
2	D3	I/O	Datenbit 3
3	D4	I/O	Datenbit 4
4	D5	I/O	Datenbit 5
5	D6	I/O	Datenbit 6
6	D7	I/O	Datenbit 7
7	CE1#	I/O	Karte aktivieren 1
8	A10	I	Adressenbit 10
9	OE#	I	Ausgang aktivieren
10	A11	I	Adressenbit 11
11	A9	I	Adressenbit 9
12	A8	I	Adressenbit 8
13	A13	I	Adressenbit 13
14	A14	I	Adressenbit 14
15	WE#	I	Schreiben aktivieren
16	IREQ#	O	Interrupt-Anforderung
17	VCC		VCC
18	VPP1		Programmspannung 1
19	MIVAL	I	MP-Eingang gültig
20	MCLK	I	MPEG-2-Takt-Eingang
21	A12	I	Adressenbit 12
22	A7	I	Adressenbit 7
23	A6	I	Adressenbit 6
24	A5	I	Adressenbit 5
25	A4	I	Adressenbit 4
26	A3	I	Adressenbit 3
27	A2	I	Adressenbit 2
28	A1	I	Adressenbit 1
29	A0	I	Adressenbit 0
30	D0	I/O	Datenbit 0
31	D1	I/O	Datenbit 1
32	D2	I/O	Datenbit 2
33	IOIS16#		16-Bit I/O (immer 'high')
34	GND		Erde

Stift	Signal	I/O	Funktion
35	GND		Erde
36	CD1#	O	Karte erkennen 1
37	MDO3	O	MP-Daten-Ausgang 3
38	MDO4	O	MP-Daten-Ausgang 4
39	MDO5	O	MP-Daten-Ausgang 5
40	MDO6	O	MP-Daten-Ausgang 6
41	MDO7	O	MP-Daten-Ausgang 7
42	CE2#	I	Karte aktivieren 2
43	VS1#	O	Spannung abfühlen 1
44	IORD#	I	I/O lesen
45	IOWR#	I	I/O schreiben
46	MISTRT	I	MP-Eingang starten
47	MD10	I	MP-Daten-Eingang 0
48	MD11	I	MP-Daten-Eingang 1
49	MD12	I	MP-Daten-Eingang 2
50	MD13	I	MP-Daten-Eingang 3
51	VCC		VCC
52	VPP2		Progr.-Spannung 2
53	MD14	I	MP-Daten-Eingang 4
54	MD15	I	MP-Daten-Eingang 5
55	MD16	I	MP-Daten-Eingang 6
56	MD17	I	MP-Daten-Eingang 7
57	MCLKO	O	MPEG-2-Takt-Ausgang
58	RESET	I	Karte zurücksetzen
59	WAIT#	O	erweiterter Buszyklus
60	INPACK#	O	Eingangsanschl.-Best.
61	REG#	I	Register auswählen
62	MOVAL	O	MP-Ausgang gültig
63	MOSTRT	O	MP-Ausgang Start
64	MDO0	O	MP-Daten-Ausg. 0
65	MDO1	O	MP-Daten-Ausg. 1
66	MDO2	O	MP-Daten-Ausg. 2
67	CD2#	O	Karte erkennen 2
68	GND		Erde

A.5.5.3 16-Bit-PC-Karten-Merkmale

Es gilt 4.3 in [6]. Es wird empfohlen, dass mindestens 12 Adressenbits während des Speicherzugriffs in dem Modul decodiert werden (4096 Bytes). Eine geringere als in der CIS (Karten-Informations-Struktur) angegebene Anzahl von Bits muss während der I/O-Zugriffe decodiert werden.

A.5.5.4 Signalbeschreibung

Es gelten 4.4, 4.6 und 4.7 von [6]. Die folgende Information ergänzt auch die Spezifikation:

Es stehen die Eingangs- und Ausgangsbusse der MPEG-2-Transportstrom-Schnittstelle zur Verfügung (MDI0-7, MDO0-7), ebenso die Steuersignale MCLKI, MCLKO, MISTRT, MIVAL, MOSTRT und MOVAL.

MCLKI läuft mit der Datenrate, mit der die Bytes dem Modul an MDI0-7 angeboten werden. MCLKO läuft mit der Datenrate, mit der die Bytes vom Modul an MDO0-7 angeboten werden. Bei Modulen, durch die der Transportstrom läuft, wird MCLKO in den meisten Fällen eine gepufferte Version von MCLKI mit einer kleinen Verzögerung sein. Bei Modulen, die Daten erzeugen, z. B. eine abnehmbare Eingangsstufe oder ein Anschluss an ein Netz, darf MCLKO von dieser Datenquelle abgeleitet werden. Bild A.18 zeigt die Beziehungen der zeitlichen Abläufe zwischen den Datensignalen, die der MPEG-2-Transportstrom-Schnittstelle und MCLKI, MCLKO zugeordnet sind, und Tabelle A.18 gibt die Grenzwerte dieser zeitlichen Abläufe an. Es ist zu beachten, dass die Spezifikation für die Grenzwerte des Zeitablaufs am Ausgang durch Gewinnen des Ausgangsignals an der Abfallflanke von MCLKO üblicherweise leicht eingehalten werden können. Für die Verzögerung zwischen MCLKI und MCLKO gibt es keine Spezifikationen. Wenn ein Modul seine eigene MCLKO liefert, darf dies nicht genau dieselbe Frequenz sein. Hauptgeräte müssen MCLKO von einem Modul zu MCLKI des nächsten Moduls der Priorität entsprechend verketten, wobei sie mehrere Buchsen der einheitlichen Schnittstellen unterstützen und die Transportstrom-Taktreferenz für das übrige Hauptgerät vom MCLKO der letzten Buchse in der Kette abgeleitet werden muss. Sowohl MCLKI als auch MCLKO müssen kontinuierlich sein. Es ist nicht vorgesehen, Burstsynchronisation einzusetzen. Diskontinuierliche Daten werden durch entsprechende Verwendung von MIVAL und MOVAL bearbeitet.

Während des ersten Bytes jedes Transportpaketes ist MISTRT an MDI0-7 gültig. Der zeitliche Ablauf an den Flanken dieses Signals muss derselbe sein wie an MDI0-7.

Tabelle A.18 – Grenzwerte der zeitlichen Beziehung

	Symbol	Min.	Max.
Taktperiode	tclkp	111 ns	
MCLKI Taktzeit 'High'	tclkh	20 ns	
MCLKI Taktzeit 'Low'	tclkl	20 ns	
MCLKO Taktzeit 'High'	tclkh	40 ns	
MCLKO Taktzeit 'Low'	tclkl	40 ns	
Eingangsdaten-Einrichten	tsu	15 ns	
Eingansdaten-Halten	th	10 ns	
Ausgangsdaten-Einrichten	tosu	20 ns	
Ausgangsdaten-Halten	toh	15 ns	

MIVAL zeigt gültige Datenbytes an MDI0-7 an. Alle Bytes eines Transportpaketes können zeitlich aufeinander folgen. In diesem Fall ist MIVAL während der gesamten Dauer des Transportpaketes logisch 1. Bestimmte in Hauptgeräten angenommene Taktstrategien können dort jedoch Lücken von einer oder mehreren Byteperioden zwischen manchen aufeinander folgenden Bytes innerhalb und/oder zwischen Transportpaketen erfordern. Dann geht MIVAL für eine oder mehrere Byteperioden auf logisch 0, um Datenbytes anzuzeigen, die ignoriert werden sollten.

MDO0-7 ist der Ausgangsbus für die MPEG-2-Transportstrom-Schnittstelle. Wenn MCLKO von MCLKI abgeleitet ist und entsprechend die Daten an MDO0-7 eine verzögerte und möglicherweise descrambelte Version der Daten an MDI0-7 sind, muss die zeitliche Beziehung zwischen den Eingangsdaten und den Ausgangsdaten entsprechend den Festlegungen in 5.4.2 der Norm geregelt werden.

Während des ersten Bytes eines Ausgangs-Transportpaketes ist MOSTRT gültig.

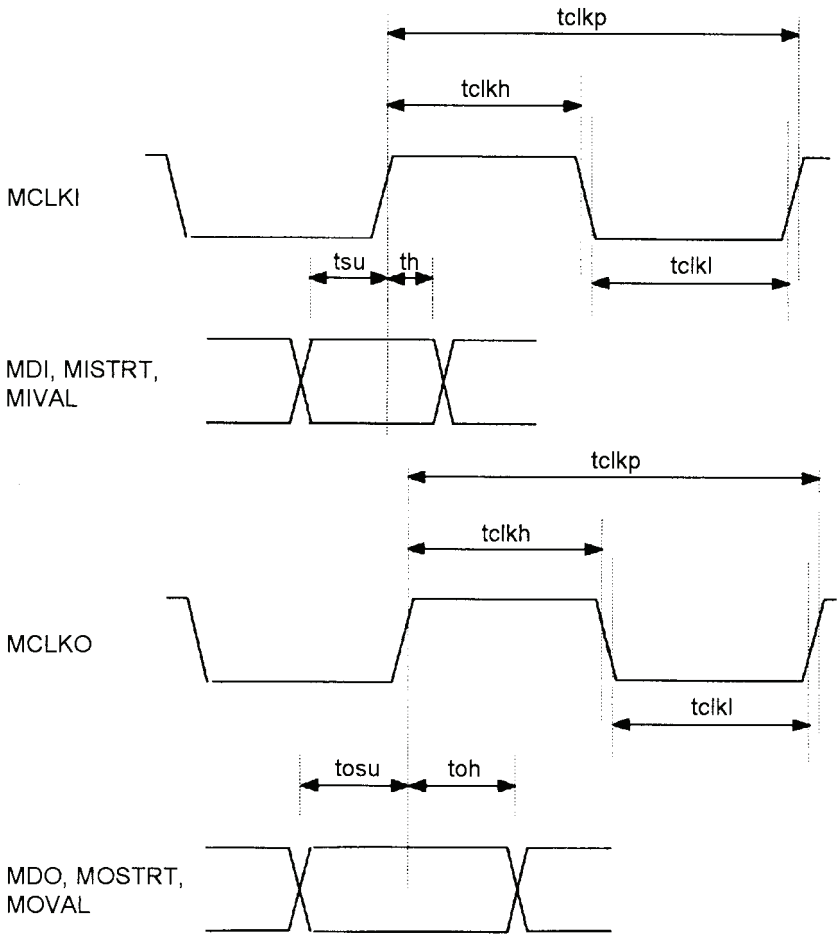

Bild A.18 – Zeitliche Abläufe für die Transportstrom-Schnittstellensignale

MOVAL zeigt die Gültigkeit von Bytes an MDO0-7 auf ähnliche Art an wie MIVAL. MOVAL braucht nicht notwendigerweise eine zeitverzögerte Version von MIVAL zu sein (siehe 5.4.2 dieser Spezifikation).

Unterstützung für Interruptanforderungen durch Module, wie in 4.4.7 von [6] definiert, ist obligatorisch. Hauptgeräte können die Interrupts benutzen oder nach Wunsch abfragen. Jedes Mal, wenn das Modul auf eine I/O-Operation antwortet, muss er INPACK# einsetzen (siehe 4.4.22 von [6]).

A.5.5.5 Speicherfunktion

Es gilt 4.6 in [6]. Eine von Hauptgeräten unterstützte Attributspeicherfunktion ist obligatorisch. Es ist zu beachten, dass der Attributespeicher byteorientiert speichert. Attributspeicherdaten treten nur an den Datenleitungen D7 bis D0 auf. Aufeinander folgende Bytes haben auch aufeinander folgende gerade Adressen (0, 2, 4 usw.). Es ist auch zu beachten, dass der Attributespeicher auch dann gelesen oder beschrieben werden kann, wenn das Modul für den Betrieb über die DVB-konforme Einheitliche Schnittstelle konfiguriert ist. Die Unterstützung der Funktion eines gemeinsamen Speichers im Hauptgerät ist freigestellt. Module dürfen keinen gemeinsamen Speicher benutzen.

A.5.5.6 Funktionen des zeitlichen Ablaufs

Es gilt 4.7 in [6]. Unterstützung des Attributspeichers durch das Hauptgerät ist obligatorisch, Unterstützung der gemeinsamen Speicher ist freigestellt.

A.5.5.7 Elektrische Schnittstelle

Die Spezifikation der einheitlichen Schnittstelle für DVB muss die in der nachstehenden Tabelle definierten logischen Schwellenspannungspegel verwenden.

Unterstützung durch Hauptgeräte für überlappende I/O-Adressenfenster, wie sie in 4.9.3.2 definiert werden, ist freigestellt. Module müssen ein 4 Bytes großes unabhängiges I/O-Adressenfenster verwenden.

Gleichspannungspegel		Hauptgerät		Karte
Parameter	Bedingungen	min.	max.	
V_{IH}	V_{CC} = 5 V 5 % V_0 = 0 V	2,0 V	V_{CC} + 0,5 V	TTL
V_{IL}	V_{CC} = 5 V 5 % V_0 = 0 V	V_0 – 0,5 V	0,8 V	TTL
V_{OH}	V_{CC} = 5 V 5 % V_0 = 0 V	2,4 V	V_{CC}	TTL
V_{OL}	V_{CC} = 5 V 5 % V_0 = 0 V	V_0	0,4 V	TTL

A.5.5.8 Karte erkennen

Es gilt 4.10 in [6].

A.5.5.9 Batteriespannung erkennen

Module dürfen diese Funktion nicht implementieren oder erfordern. Ihre Unterstützung durch das Hauptgerät ist freigestellt.

A.5.5.10 Anschalten und Abschalten

Es gilt 4.12 in [6], einschließlich der Spezifikation für den mittleren Strom während der Konfiguration. Der Betriebsstrom des Moduls muss in dessen CIS (3.3.2 von [8]) angegeben werden. Jedes Modul darf nicht mehr als 1,5 W verbrauchen. Zusätzlich darf der Strom der Stromversorgung zu jedem Modul (Summe des Stromes von V_{cc} und V_{pp}) als Langzeitwert 300 mA nicht übersteigen und der Kurzzeit-Spitzenstrom ist für kürzer als 1 ms auf 500 mA begrenzt. Hauptgeräte brauchen alternative Werte von V_{pp} nicht zu unterstützen. Wenn sie es nicht tun, muss es dieselbe Spannung sein wie V_{cc}.

A.5.5.11 I/O-Funktion

Es gilt 4.13 in [6] mit der Ausnahme, dass Module nur 8-Bit-Lese- und Schreibmodes benutzen dürfen.

A.5.5.12 Konfiguration der Funktionen

Es gilt 4.14 in [6]. Module müssen nur das Konfigurationsauswahlregister unterstützen. Die Unterstützung anderer Register als das Konfigurationsauswahlregister durch Hauptgeräte ist freigestellt.

A.5.5.13 Konfiguration der Karten

Es gelten 4.15 und 4.15.1 in [6].

A.5.6 Spezifikation des Metaformates

Dieser Abschnitt definiert einen Mindestsatz von Tupeln, die ein Modul mit Einheitlicher Schnittstelle in seiner Karten-Informationsstruktur haben muss. Dies ist auch der Mindestsatz, den ein Hauptgerät braucht, um erkennen zu können.

Es gelten nur die Abschnitte 1, 2 und 3 von [8].

Die folgenden Tupeln sind der Mindestsatz, der bei Modulen mit Einheitlicher Schnittstelle benutzt wird, und der alleinige Satz, der erforderlich ist, um von Hauptgeräten erkannt zu werden. Die durch sie gebildete Tupelkette muss die erste im Attributspeicher sein, obwohl ihr weitere folgen können, für die es nicht erforderlich ist, vom Hauptgerät erkannt zu werden. Das Hauptgerät muss die Möglichkeit berücksichtigen, dass auf diese Tupel weitere Tupeln folgen können. Die Kette muss durch einen Tupel CISTPL_NOLINK beendet werden. Alle nachstehend gegebenen Verweisungen auf Abschnitte beziehen sich auf [8].

1 CISTPL_DEVICE: Codiert nach 3.2.2 und 3.2.4.

2 CISTPL_DEVICE_A: Codiert nach 3.2.2 und 3.2.4.

3 CISTPL_DEVICE_OC: Codiert nach 3.2.3 und 3.2.4.

4 CISTPL_DEVICE_OA: Codiert nach 3.2.3 und 3.2.4.

5 CISTPL_VERS_1: TPLLV1_MAJOR = 05h; TPLLV1_MINOR = 00h; weitere vom Modulhersteller definierte Felder nach 3.2.10.

6 CISTPL_MANFID: Vom Hersteller nach 3.2.9 definiert.

7 CISTPL_CONFIG: Vom Hersteller nach 3.3.4 definiert. Die Basisadresse des Konfigurationsregisters darf nicht größer sein als FFEh. In dem Feld TPCC_SBTPL muss es einen Subtupel geben (siehe 3.3.4.5). In diesem Subtupel muss STCI_TFN die ID-Nummer 0241h enthalten, die für diese Klasse von Modulen von PCMCIA ausgegeben wird, und STCI_STR muss eine einzelne Folge 'DVB_CI_V1.00' enthalten. Der Wert von STCI_IFN ist die spezielle Anzeige, dass dies ein zur DVB-konformen einheitlichen Schnittstelle passendes Modul ist. Die STCI_STR-Folge zeigt die Versionsnummer an. Die letzten 5 Zeichen dieser Folge werden immer 'Vx.xx' sein, dabei ist x ein Digit und zeigt die Spezifikationsversion an, zu der das Modul passt.

8 CISTPL_CFTABLE_ENTRY: Festgelegt nach 3.3.2. Module müssen mindestens einen dieser Einträge unterstützen. Für den ersten Eintrag hat TPCE_INDEX sowohl Bit 6 (Vorgabe) als auch Bit 7 (Schnittstelle) gesetzt. Die Nummer des Konfigurationseintrags kann jeder passende Wert ungleich Null sein. TPCE_IF = 04h zeigt die Kundenschnittstelle 0. TPCE_FS muss das Vorhandensein von I/O-, Stromversorgungs- und IRQ-Konfigurationseinträgen anzeigen. TPCE_IO ist ein 1-Byte-Feld mit dem Wert 22h (siehe 3.3.2.6). Die Information bedeutet: 2 Adressenzeilen werden durch das Modul decodiert und es benutzt dabei nur 8-Bit-Zugriffe. Der Stromversorgungs-Konfigurationseintrag, der von A.5.5.10 dieser Spezifikation gefordert wird, muss der PC-Karten-Spezifikation entsprechen [8]. TPCE_IR ist gleich 20h (siehe 3.3.2.7). Die Information bedeutet: Interruptanforderungen liegen im Pegelmode am Kontaktstift für Interruptanforderungen und dieser Kontaktstift kann bei aktiven Low-Pegel aktiv bleiben, bis der Interrupt bedient wird, und die Bytes 2 und 3 der TPCE_IR-Information nicht bereitgestellt werden. Der CF-Tabelleneintrag enthält auch die beiden folgenden Untertupels:

9 STCE_EV: siehe 3.3.2.10.1. Es ist nur der Systemname 'DVB_HOST' vorhanden.

10 STCE_PD: siehe 3.3.2.10.2. Es ist nur der Name 'DVB_CI_MODULE' der physikalischen Baueinheit vorhanden.

11 CISTPL_END: der Wert FFh. Wenn das CA-Modul zusätzlich zu den oben definierten weitere Tupel enthält, dann kommen diese vor CISTPL_END.

Anhang B
(informativ)

Zusätzliche Objekte

B.1 Authentisierung

Dieses wahlfreie Hilfsmittel ist in Hauptgeräten enthalten, die Authentisierung unterstützen, um die Übertragung von Authentisierungsbefehlen zwischen einem abnehmbaren Modul und einem Hauptgerät so zu ermöglichen, dass die Benutzung jedes Authentisierungsverfahrens unter Leitung eines CA-System-Operators nur bezüglich der von ihm gesteuerten Signale erfolgt. Ein Rundfunkbetreiber und/oder CA-System-Operator, der das Authentifizierungsprotokoll nicht benutzen will, muss in der Lage sein, Signale zu übertragen, die die Einheitliche Schnittstelle ohne Eingriff in irgendeinen Authentisierungsvorgang durchlaufen.

Das Hilfsmittel besteht aus zwei Objekten, Authentisierungsanforderung und Authentisierungsantwort, die benutzt werden, um ein Authentisierungsprotokoll zu implementieren. Die spezielle Verwendung wird zwischen CA-Herstellern und Herstellern von Hauptgeräten definiert, die dieses Hilfsmittel enthalten und wird hier aber nicht definiert.

B.1.1 Authentisierungsanforderung (auth_req) und Authentisierungsantwort (auth_resp)

Diese Objekte sind mit Ausnahme des Tag-Wertes identisch.

Syntax	Anzahl der Bits	Mnemonik
auth_req () {		
auth_req_tag	**24**	**uimsbf**
length_field()		
auth_protocol_id	**16**	**uimsbf**
for (i=0; i<n; i++) {		
auth_req_byte	**8**	**uimsbf**
}		
}		

Syntax	Anzahl der Bits	Mnemonik
auth_resp () {		
auth_resp_tag	**24**	**uimsbf**
length_field()		
auth_protocol_id	**16**	**uimsbf**
for (i=0; i<n; i++) {		
auth_resp_byte	**8**	**uimsbf**
}		
}		

B.1.2 Codierung des Hilfsmittels Authentisierung

Hilfsmittel	Klasse	Typ	Version	Hilfsmittelkennung
Authentisierung	16	1	1	00100041

apdu_tag	Tag-Wert hex	Hilfsmittel	Richtung Hauptgerät ↔ Anwendung
T_{auth_req}	9F 82 00	Authentisierung	←
T_{auth_resp}	9F 82 01	Authentisierung	→

B.2 Hilfsmittel EBU-Teletextanzeige

Das Hilfsmittel definiert ein Objekt und ein Protokoll für die Kommunikation mit dem Hauptgerät, wenn dies signalisiert, dass das Hilfsmittel EBU-Teletext(EBT)-Anzeige vorhanden ist.

EBT ist ein Mittel zum Ausgeben textorientierter Daten zur Anzeige auf dem Bildschirm eines Fernsehgerätes. Es wird üblicherweise in den Reservezeilen der Bildaustastlücke von konventionellen Fernsehsignalen transportiert und wird in den Transportströmen der Multiplexsignale von MPEG-2-Sendungen übertragen. ETB definiert eine spezielle Anzeigemöglichkeit in dem Fernsehgerät. Die folgende Reihe von Objekten wird definiert, um die Daten für die Anzeige zu übermitteln, die diese Möglichkeit nutzt. Die Gesamtspezifikation wird in [B1] angegeben, aber die herausragenden Merkmale für Anzeigezwecke werden hier wiederholt.

B.2.1 Anzeigekennwerte

Die Anzeige besteht aus einem zweidimensionalen Alphamosaikzeichen-Gitternetz aus 40 bis 160 Zeichen je Zeile und bis zu 101 Zeilen. Bei Anzeigen mit dem Bildseitenverhältnis 4:3 und 625 Zeilen wird dies auf 40 Zeichen je Zeile und 25 Zeilen beschränkt. Die Spezifikation verwendet einen definierten Zeichensatz für die Anzeige, sie lässt aber auch frei definierbare Zeichensätze (DRCS) zu. Sie verwendet für die Anzeige einen Standardsatz von Farben, lässt aber auch eine Zusammenstellung von verschiedenen herunterzuladenden Farbtabellen zu. So wie eine Alphamosaik-Anzeigemöglichkeit stellt die Norm auch eine Alphageometrische- und eine Alphaphotographische-Anzeige zur Verfügung. Die augenblickliche Version dieses Hilfsmittels unterstützt diese aber nicht. Die Anzeige muss der Darstellungsebene-2-Syntax akzeptieren.

B.2.2 Objekt Kommunikationsphilosophie

Die hier angenommene Kommunikationsphilosophie muss die EBT-Anzeige mit Daten versorgen, so wie sie terrestrisch, über Kabel oder in einem MPEG-2-Transportstrom übertragen wurden. Für diesen Zweck wird ein Objekt EBT-Pakete zur Verfügung gestellt. Es wird eine Sitzung für das Hilfsmittel EBT-Anzeige eingerichtet, was gelingt, wenn keine andere Anwendung dieses Hilfsmittel benutzt. Solange die Anwendung die Steuerung ausübt, wird jedem eingehenden Teletextdienst der Zugang zur Anzeige verweigert. Die Anzeige wird auf der Basis 'wer zuerst kommt, mahlt zuerst' der Anwendung zugewiesen. Nachdem das erste Objekt EBT-Pakete empfangen wurde, muss die Anzeige von Bild- auf Textmode umschalten und anzeigen, was geladen wurde. Weitere Objekte EBT-Pakete können, wenn erforderlich, gesendet werden. Beim Löschen der Sitzung muss die Anzeige in den Bildmode zurückkehren.

B.2.3 EBT-Pakete

Syntax	Anzahl der Bits	Mnemonik
ebt_packets() {		
ebt_packet_tag	**24**	**uimsbf**
length_field()		
for (i:=0;i<no_of_packets; i++) {		
for (j=0; j<42; j++) {		
ebt_packet_byte	**8**	**uimsbf**
}		
}		
}		

Die Daten in einem Objekt EBT-Pakete bestehen aus mehreren Zeilen, von denen jede 42 Bytes lang ist und dem Format für 625-Zeilen-Anwendungen in der EBT-Spezifikation entspricht. In jeder Zeile enthalten die ersten zwei Bytes die Magazin- und Paket-Adressenfelder. Jedes Byte ist mit 4 Informationsbits und 4 Paritätsbits hammingcodiert. Von den acht Informationsbits in den beiden Bytes bilden die ersten drei Bits die Magazinnummer (X) und die letzten 5 Bits die Paketadresse (Y). Durch Vereinbarung liegt X im Bereich von 1 bis 8, wobei 8 durch 000 dargestellt wird. Y liegt im Bereich von 0 bis 31.

Die Codierung der weiteren 40 Bytes hängt von dem Wert des X- und Y-Feldes ab. Im Allgemeinen sind Pakete mit Y = 0 Headerpakete und enthalten weitere hammingcodierte Information, auf die Anzeigetext folgt. Pakete mit Y = 1 bis 25 enthalten 40 Bytes Text- und Anzeigesteuerungszeichen zur Anzeige auf dem Bildschirm. Pakete mit Y = 26 bis 31 übertragen im Allgemeinen Hilfs-Anzeige- und Steuerungsinformation zur Verbesserung der Eigenschaften.

B.2.4 Codierung des Hilfsmittels EBU-Teletext

Hilfsmittel	Klasse	Typ	Version	Hilfsmittelbezeichner
EBU-Teletext	128	1	1	00800041

apdu_tag	Tag-Wert (hex)	Hilfsmittel	Richtung Hauptgerät↔Anwendung
$T_{ebt\_packets}$	9F 90 00	EBU-Teletext	←

B.2.5 Verweisungen

[B1] EBU SPB 492: Teletext Spezifikation (625 line television systems), EBU, Genf.

B.3 Hilfsmittelklasse Chipkartenleser

Dieses Schriftstück beschreibt ein wahlfreies Hilfsmittel Chipkartenleser, das direkt vom Hauptgerät oder einem anderen Modul zur Verfügung gestellt werden kann. Dieses Hilfsmittel benutzt Befehle auf der Karten-Befehlsebene (Header, Daten und Statusbytes). Die Befehle werden in den Spezifikationen ISO 7816-1 bis ISO 7816-3 definiert. Es werden vier Objekte definiert. Smard Card Cmd (Chipkartenbefehl) und Smard Card Reply (Chipkartenantwort) führen Steuer- und Antwortfunktionen aus. Smard Card Send (Chipkarte Senden) und Smard Card Rcv (Chipkarte Empfangen) senden und empfangen Daten. Mit diesem Hilfsmittel können mehrere Sitzungen eingerichtet werden, so dass verschiedene Anwendungen das Hilfsmittel abfragen können, es kann aber immer nur eine Sitzung im Zustand 'verbunden' sein.

Das Hilfsmittel ist dazu bestimmt, den Chipkartenleser im Hauptgerät oder in einem Modul für kurzzeitige Sitzungen wie für Bankkarten-, Teleshopping- oder pay-per-view-Transaktionen zu unterstützen. Um die Zugriffsmöglichkeiten auf einen gescrambelten Dienst aufrechtzuerhalten, wird empfohlen, das Hilfsmittel nicht für Langzeitsitzungen mit Chipkarten zu benutzen, die für laufende Sicherheitsoperationen erforderlich sind, weil nämlich dadurch das Hilfsmittel Chipkartenleser blockiert wird und für andere Verwendung nicht verfügbar ist und weil auch keine Garantie für eine begrenzte Antwortverzögerung gegeben werden kann.

B.3.1 Objekte

B.3.1.1 Chipkartenbefehl (Smard Card Cmd)

Syntax	**Anzahl der Bits**	**Mnemonik**
smart_card_cmd () {		
smart_card_cmd_tag	**24**	**uimshf**
length_field ()		
card_interface_cmd_id	**8**	**uimsbf**
}		

smard_card_cmd_id	Chipkartenbefehl-Kennung	Kennungswert
connect	Verbinden	01
disconnect	Trennen	02
power_on_card	Karte Anschalten	03
power_off_card	Karte Abschalten	04
reset_card	Karte Zurücksetzen	05
read_status	Kartenstatus Lesen	06
read_answer _to_reset	Antwort auf Rücksetzen Lesen	07
reserved	Reserviert	Andere Werte

Dieser Befehl wird von der Anwendung ausgegeben, sobald eine Sitzung für das Hilfsmittel eingerichtet werden muss. Als Antwort wird von dem Hilfsmittel ein Objekt Chipkartenantwort gegeben. Die folgenden Befehle sind vorhanden:

– **connect:** verbindet diese Sitzung mit dem Kartenleser. Wenn die Verbindung erfolgreich ist, dann ist die Antwort 'connected' mit einem entsprechenden Kartenstatus card_inserted, no_card usw. Wenn der Kartenleser schon in einer anderen Sitzung verbunden ist, dann wird 'busy' geantwortet.

– **disconnect:** trennt den Kartenleser von dieser Sitzung. Die Stromversorgung der Karte wird abgeschaltet. Mit dem entsprechenden Kartenstatus wird die Antwortkennung 'free' geantwortet. Wenn der Kartenleser mit einer anderen Sitzung verbunden ist, dann wird 'busy' geantwortet.

– **power_on_card:** schaltet die Kartenschnittstelle durch Verbinden und Aktivieren der Kontakte durch die Schnittstelleneinrichtung ein (V_{cc} dann RAZ nach Spezifikation ISO 7816). Der Befehl setzt auch die Karte zurück. Der Befehl ist nur zulässig, wenn der Kartenleser mit dieser Sitzung verbunden ist. Die Antwort ist 'free', wenn der Kartenleser nicht verbunden ist, 'busy', wenn er in einer anderen Sitzung verbunden ist, 'answ_to_reset', wenn eine Karte eingeschoben ist und ein Rücksetzen korrekt durchführt, und 'no_answ_to_reset', wenn keine Karte vorhanden ist oder die Karte nicht korrekt zurücksetzt.

– **power_off-card:** schaltet die Kartenschnittstelle nach der Spezifikation ISO 7816 ab. Die Antwort ist 'connected', wenn die Sitzung verbunden und der Vorgang erfolgreich verlaufen ist, 'free', wenn die Sitzung nicht verbunden ist, und 'busy', wenn eine andere Sitzung verbunden ist.

– **reset_card:** setzt die Kartenschnittstelle zurück, wenn die Chipkarte eingeschaltet ist. Die Antworten sind dieselben wie für 'power_on_card'.

– **read status:** liest den augenblicklichen Verbindungs- und den Kartenstatus. Es ist nicht nötig zu verbinden, um diesen Befehl auszugeben.

– **read_answ_to_reset:** gibt, wenn eine Karte eingesteckt ist, die Nachricht 'Antwort auf Rücksetzen' für die augenblickliche Karte. Es ist nicht nötig zu verbinden, um diesen Befehl auszugeben. Die Antwort ist 'answ_to_reset', wenn eine Karte vorhanden ist, andernfalls 'no_answ_to_reset'.

B.3.1.2 Chipkartenantwort (Smard Card Reply)

Syntax	Anzahl der Bits	Mnemonik
smart_card_reply() {		
smart_card_reply_tag	**24**	**uimsbf**
length_field ()		
smart_card_reply_id	**8**	**uimsbf**
smart_card_status	**8**	**uimsbf**
if (smart_card_reply_id == answ_to_reset) {		
for (i=0; i < n; i++) {		
char_value	**8**	**uimsbf**
}		
}		
}		

smard_card_reply_id	Chipkartenantwort-Kennung	Kennungswert
connected	Verbunden	01
free	Frei	02
busy	Belegt	03
answ_to_reset	Antwort auf Rücksetzen	04
no_answ_to_reset	Keine Antwort auf Rücksetzen	05
reserved	Reserviert	Andere Werte

smard_card_status	Chipkartenstatus	Wert
card_inserted	Karte eingeschoben	01
card_removed	Karte herausgenommen	02
card_in_place_power_off	Karte auf Platz, Stromversorgung aus	03
card_in_place_power_on	Karte auf Platz, Stromversorgung ein	04
no_card	Keine Karte	05
unresponsive_card	Nicht antwortende Karte	06
refused_card	Abgelehnte Karte	07
reserved	Reserviert	Andere Werte

Die Chipkartenantwort wird als Antwort auf ein Objekt Chipkartenbefehl gegeben oder als Antwort auf ein Objekt Chipkarten Senden, wenn Senden in diesem Augenblick eine ungeeignete Operation ist. Sie wird auch unangefordert an die gegebenenfalls angeschlossene Sitzung gesendet, wenn eine Karte entfernt oder eingesteckt wurde, um diese Tatsache zu signalisieren. Die Antwort-ID-Werte sind:

– **connected:** ist die Antwort auf eine angeschlossene Sitzung, bei der answ_to_reset oder no_answ_to_reset nicht gesendet zu werden braucht.
– **free:** wird als Antwort auf einen Befehl 'disconnect' gesendet oder auf andere Befehle, wenn der Chipkartenleser nicht angeschlossen ist.
– **busy:** wird als Antwort auf nicht erlaubte Befehle an einen Chipkartenleser gesendet, der schon mit einer anderen Sitzung verbunden ist.
– **answ_to_reset:** wird mit der Nachricht Antwort auf Rücksetzen-Nachricht von der Karte gesendet.
– **no_answ_to_reset:** wird gesendet, wenn Antwort auf Rücksetzen angefordert wurde, jedoch nicht verfügbar ist.

Jede Antwort überträgt auch den Kartenstatus. Die folgenden Statuswerte werden benutzt:

– **card_inserted:** wird unaufgefordert an eine angeschlossene Sitzung gesendet, wenn eine Karte eingeschoben ist.
– **card_removed:** wird unaufgefordert an eine angeschlossene Sitzung gesendet, wenn eine Karte entnommen ist.
– **card_in_place_power_off:** wird in allen Antworten gesendet, wenn eine Karte eingesteckt und erkannt und die Stromversorgung der Karte nicht eingeschaltet ist.
– **card_in_place_power_on:** wird in allen Antworten gesendet, wenn eine Karte eingesteckt und erkannt und die Stromversorgung der Karte eingeschaltet ist.
– **no_card:** wird in allen Antworten gesendet, wenn keine Karte eingeschoben ist.
– **unresponsive_card:** in allen Antworten, wenn die eingeschobene Karte auf ein Rücksetzen nicht antwortet oder wenn sie auf ein Chipkarten Senden nicht antwortet.
– **refused_card:** wird in allen Antworten gesendet, nachdem eine Karte auf ein Rücksetzen geantwortet hat, aber nicht in der erwarteten Art.

B.3.1.3 Chipkarte Senden (Smart Card Send)

Syntax	**Anzahl der Bits**	**Mnemonik**
smart_card_send () {		
smart_card_send_tag	**24**	**uimsbf**
length_field()		
CLA	**8**	**uimsbf**
INS	**8**	**uimsbf**
P1	**8**	**uimsbf**
P2	**8**	**uimsbf**
length_in	**16**	**uimsbf**
for (i=0; i<length_in; i++)		
char_value	**8**	**uimsbf**
}		
length_out	**16**	**uimsbf**
}		

Chipkarte Senden wird benutzt, um Daten an eine angeschlossene Karte zu senden, die einen Rücksetzzyklus abgeschlossen hat. Die Antwort ist entweder ein Objekt Chipkarte Empfangen mit Daten und/oder die Statusbytes SW1 und SW2 oder eine Chipkartenantwort mit Statusinformation, wenn es nicht passend ist in diesem Augenblick Daten zu der Karte zu senden.

B.3.1.4 Chipkarte Empfangen (Smart Card Rcv)

Syntax	Anzahl der Bits	Mnemonik
smart_card_rcv () {		
smart_card_rcv_tag	**24**	**uimsbf**
length_field ()		
for (i=0; i < n; i++) {		
char_value	**8**	**uimsbf**
}		
SW1	**8**	**uimsbf**
SW2	**8**	**uimsbf**
}		

Chipkarte Empfangen wird als Antwort auf Chipkarte Senden an eine angeschlossene Sitzung mit einer in Betrieb befindlichen Karte gesendet.

B.3.2 Codieren des Hilfsmittels Chipkartenleser

Hilfsmittel	Klasse	Typ	Version	Hilfsmittelbezeichner
Chipkartenleser	112	siehe B.3.2.1	1	00700041

apdu_tag	Tag-Wert (hex)	Hilfsmittel	Richtung Hauptgerät ↔ Anwendung
$T_{smard\_card\_cmd}$	9F 8E 00	Chipkartenleser	←
$T_{smard\_card\_reply}$	9F 8E 01	Chipkartenleser	→
$T_{smard\_card\_send}$	9F 8E 02	Chipkartenleser	←
$T_{smard\_card\_rcv}$	9F 8E 03	Chipkartenleser	→

B.3.2.1 Codierung des Hilfsmitteltyps

Das Hilfsmittel Chipkartenleser kann in bis zu 16 Einzelfällen eingesetzt werden, die entweder vom Hauptgerät oder von anderen Modulen veranlasst werden. Die Anwendung wählt den benötigten Kartenleser durch Angeben der erforderlichen Baueinheitennummer, wie in Bild B.1 gezeigt, in Form der vier niedrigstwertigen Bits des Feldes Hilfsmitteltyp aus.

Bit	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	Baueinh. Nr.			

Bild B.1 – Struktur des Hilfsmitteltyps Chipkartenleser

B.4 Klasse für die Unterstützung zukünftiger DVB-EPG-Ereignisse

Dieses Hilfsmittel wird durch eine EPG-Anwendung zur Verfügung gestellt, um Kommunikation von Berechtigungsinformationen von CA-Modulen an elektronische Programmführer – entweder auf einem Hauptgerät oder Modul basierend – zu unterstützen. CA-Module, die Berechtigungsinformation für EPG-Anwendungen anbieten, müssen eine Sitzung mit allen Einzelfällen dieses Hilfsmittels einrichten, die im Hilfsmittelmanager angezeigt sind und müssen das von der EPG-Anwendung angeforderte Protokoll abwickeln.

Das Protokoll verwendet zwei Objekte – ein Objekt event_enquiry (Ereignisanfrage) und ein Objekt event_reply (Ereignisantwort). Das Protokoll erlaubt einem EPG den Berechtigungsstatus eines laufenden oder zukünftigen

Ereignisses anzufragen, das durch seine Ereignis-ID in der Dienstinformation gekennzeichnet ist. Der CA-Modul kann antworten und zeigt damit an, dass die Berechtigung für das Ereignis verfügbar ist, nicht verfügbar ist, nach Dialog verfügbar sein kann (z. B. pay per view) oder dass sein Berechtigungsstatus unbekannt ist. Zusätzlich unterstützt das Protokoll einen Prozess, der dem CA-Modul erlaubt, einen Dialog zu starten, um eine mögliche Berechtigung verfügbar zu machen, z. B. für pay per view und dann zu dem EPG-Dialog zurückzukehren. Dieses Hilfsmittel verhindert nicht EPG- und CA-Anwendung von demselben oder zusammenarbeitenden Anbietern, die einen privaten Hilfsmittelmechanismus benutzen, um den Betrieb von EPG und CA besser zu integrieren. Ein CA-Modul jedoch, das Berechtigungsinformation zu unabhängigen EPGs liefern kann, muss dieses Hilfsmittel benutzen.

B.4.1 Objekte

B.4.1.1 Objekt Ereignisabfrage

Dies ist eine Anfrage vom EPG beim CA-Modul.

Tabelle B.1 – Codierung des Objektes Ereignisanfrage

Syntax	Anzahl der Bits	Mnemonik
event_enquiry() {		
event_enquiry_tag	**24**	**uimsbf**
length_field()		
event_cmd_id	**8**	**uimsbf**
network_id	**16**	**uimsbf**
original_network_id	**16**	**uimsbf**
transport_stream_id	**16**	**uimsbf**
service_id	**16**	**uimsbf**
event_id	**16**	**uimsbf**
}		

event_cmd_id (Ereignis-Befehl-Kennung)

Dieser kann die folgenden Werte annehmen:

event_cmd_id	Ereignis-Befehl-Kennung	Wert
mmi	Mensch-Maschine-Schnittstelle (MMI)	02
query	Abfrage	03
reserved	Reserviert	Andere Werte

Der 'MMI'-Befehl ermöglicht dem CA-Modul, einen MMI-Dialog zu beginnen, um eine Berechtigung bereitzustellen. Er wird typischerweise benutzt, nachdem der EPG schon bestimmt hat (durch einen vorhergehenden 'Abfrage'befehl), dass eine MMI-Sitzung durch das CA-Modul angefordert wird und seine eigene MMI-Sitzung zeitweise geschlossen hat, um das Hilfsmittel MMI für das Modul bereitzustellen. Der 'Abfrage'befehl wird benutzt, um nur nach dem augenblicklichen Berechtigungsstatus eines Ereignisses zu fragen, ohne einen Dialog anzufordern.

Die Ereignis-Kennung ist durch die Kennung an den anderen Stellen in der SI völlig bestimmt, was sie eindeutig macht.

B.4.1.2 Objekt Ereignisantwort

Dies ist eine Antwort vom CA-Modul zum EPG.

Tabelle B.2 – Codierung des Objektes Ereignisantwort

Syntax	Anzahl der Bits	Mnemonik
event_reply() {		
event_reply_tag	**24**	**uimsbf**
length_field()		
event_status	**8**	**uimsbf**
}		

event_status (Ereignisstatus)

Dieser kann die folgenden Werte annehmen:

event_status	Ereignisstatus	Wert
entitlement_unknown	Berechtigung unbekannt	00
entitlement_available	Berechtigung verfügbar	01
entitlement_not_available	Berechtigung nicht verfügbar	02
mmi_dialogue_required	MMI-Dialog erforderlich	03
mmi_complete_unknown	MMI abgeschlossen unbekannt	04
mmi_complete_available	MMI abgeschlossen verfügbar	05
mmi_complete_not_available	MMI abgeschlossen nicht verfügbar	06
reserved	Reserviert	Andere Werte

'entitlement_unknown' bedeutet, dass das CA-Modul den Berechtigungsstatus dieses Ereignisses nicht bestimmen kann. 'entitlement_available' bedeutet, dass eine Berechtigung für dieses Ereignis im Augenblick verfügbar ist. 'entitlement_not_available' bedeutet, dass eine Berechtigung für dieses Ereignis im Augenblick nicht verfügbar ist und nicht durch irgendeinen Benutzerdialog mit dem CA-Modul verfügbar gemacht werden kann. 'mmi_dialogue_required' zeigt an, dass eine Berechtigung im Augenblick nicht verfügbar ist, aber nach einem Dialog zwischen CA und Benutzer verfügbar gemacht werden kann. 'mmi _complete_unknown' wird als Antwort auf eine 'event_enquiry' mit dem MMI-Befehl gegeben, nachdem der CA-MMI-Dialog abgeschlossen ist und der Berechtigungsstatus noch unbekannt ist. 'mmi_complete_available' wird als Antwort auf eine 'event_enquiry' mit dem 'MMI'-Befehl gegeben, nachdem der CA-Dialog abgeschlossen ist und die Berechtigung gegeben wurde. 'mmi_complete_not_available' wird auf eine 'event_enquiry' mit dem 'mmi'-Befehl als Antwort gegeben, nachdem der CA-MMI-Dialog abgeschlossen ist und die Berechtigung nicht gegeben wurde, z. B. wenn der Benutzer in dem Dialog entschied, schließlich das Ereignis doch nicht zu kaufen.

Es ist zu beachten, dass sich das zurückgesandte Objekt event_reply auf das vorangegangene empfangene Objekt event_enquiry bezieht, so dass der EPG kein weiteres Objekt event_enquiry senden muss, bis er die event_reply für die laufende Anfrage empfängt.

B.4.1.3 Anwendungsbeispiel

Es wird ein typischer Protokollaustausch beschrieben. Dieser wendet eine Erweiterung der MMI-Hilfsmittel an, die es der Anwendung erlaubt, eine MMI-Sitzung mit einer kurzen Verzögerung zu schließen, bevor sie zu dem laufenden Dienst zurückgeht, um es einer weiteren Anwendung zu ermöglichen, ihren eigenen Dialog nahtlos zu dem vorhergehenden zu starten.

EPG:	Ereignis-Abfrage	(event_query)	(Abfrage, Ereignis x)
CA:	Ereignis-Antwort	(event_reply)	(MMI-Dialog angefordert)
EPG:	MMI schließen	(close_mmi)	(Verzögerung); Kontext merken
EPG:	Ereignis-Abfrage	(event_query)	(MMI, Ereignis x)
CA:	MMI eröffnen	(open_mmi)	Dialog durchführen
CA:	MMI schließen	(close_mmi)	(Verzögerung)
CA:	Ereignis-Antwort	(event_reply)	(MMI abgeschlossen verfügbar) (kann auch nicht verfügbar oder unbekannt antworten)
EPG:	MMI eröffnen	(open_mmi)	EPG-Dialog vom gemerkten Kontext fortsetzen

B.4.2 Codierung des Hilfsmittels Unterstützung zukünftige EPG-Ereignisse

Hilfsmittel	Klasse	Typ	Version	Hilfsmittelbezeichner
Unterstützung zukünftiger EPG-Ereignisse	120	siehe B.4.2.1	1	00780041

apdu_tag	Tag-Wert (hex)	Hilfsmittel	Richtung Hauptgerät ↔ Anwendung
$T_{event\_enquiry}$	9F 8F 00	Unterst. zuk. EPG-Ereig.	→
T_{event_reply}	9F 8F 01	Unterst. zuk. EPG-Ereig.	←

B.4.2.1 Codierung des Hilfsmitteltyps

Es kann sein, dass es in einem Hauptgerät mehr als einen EPG gibt. Zum Beispiel darf das Hauptgerät eine interne EPG-Anwendung unterstützen und der Benutzer darf einen gekauften, auf einem Modul basierenden EPG haben. In diesem Fall muss das Hauptgerät Hilfsmittel-Typnummern zuweisen, bei Null beginnend, um die Einzelfälle des EPG-Betriebes zu unterscheiden. CA-Module, die unter Verwendung dieses Hilfsmittels EPGs unterstützen, müssen zu jedem Einzelfall dieser Klasse, der als unterschiedlicher Hilfsmitteltyp vom Hilfsmittelmanager angekündigt wurde, eine Sitzung einrichten.