GY/Z

中华人民共和国广播电影电视行业标准化指导性技术文件

GY/Z 175—2001

数字电视广播条件接收系统规范

Specification of conditional access system for digital television broadcasting

2001-09-18 发布

国家广播电影电视总局 发布

前　言

本指导性技术文件主要参考了 DVB 系列规范，并结合了国际和国内有关条件接收的生产、开发、试验和应用的情况，从我国的国情出发编制的。

本指导性技术文件由正文和 9 个附录构成，正文给出了条件接收系统的系统构成和总体要求，在附录 A 中规定了节目信息管理系统与条件接收系统接口和要求、附录 B 中规定了用户管理系统的要求及其接口、附录 C 中规定了智能卡及其与接收机接口的要求、附录 D 中规定了条件接收智能卡标识；同时在附录 E 中规定了同密技术的规范、附录 F 中规定了多密技术的规范、附录 G 中规定了多密技术的使用、附录 H 中规定了通用加扰系统的描述、附录 I 中给出了与 CA 有关的业务信息/节目特定信息(SI/PSI)中关于条件接收的规定。

本指导性技术文件仅供参考。有关本指导性技术文件的建议和意见，向全国广播电视标准化技术委员会反映。

本指导性技术文件的附录 A、B、C、D、E、F、H 和 I 为文件的附录。

本指导性技术文件的附录 G 为提示的附录。

本指导性技术文件由全国广播电视标准化技术委员会归口。

本指导性技术文件起草单位：国家广播电影电视总局数字(高清晰度)电视标准工作组。

本指导性技术文件主要起草人：曾庆军、关宏超、白慧生、曹青、邓向冬、谷晓军、柯进、刘世贵、沈彤、孙功宪、汪莉、谢锦辉、薛滨、杨庆华、俞迁如、曾学文、赵翮。

中华人民共和国广播电影电视行业标准化指导性技术文件

数字电视广播条件接收系统规范

GY/Z 175—2001

Specification of conditional access system for digital television broadcasting

1 范围

本指导性技术文件规定了地面、有线、卫星等数字电视广播系统中条件接收系统的系统构成、总体要求，并在附录中给出了节目信息管理系统与条件接收系统接口和要求、用户管理系统的要求及其接口、智能卡及其与接收机接口的要求、条件接收智能卡标识、同密技术、多密技术、多密技术的使用、通用加扰系统和与 CA 有关的业务信息/节目特定信息（SI/PSI）的规定等。

本指导性技术文件适用于地面、有线、卫星等数字电视广播系统中的条件接收系统。

2 引用标准和文件

下列标准和文件所包含的条文，通过在本指导性技术文件中引用而构成为本指导性技术文件的条文。本指导性技术文件出版时，所示版本均为有效。所有标准和文件都会被修订，使用本指导性技术文件的各方应探讨使用下列标准和文件最新版本的可能性。

GB/T 17975.1—2000 信息技术 运动图像及其伴音信息的通用编码 第 1 部分 系统

GB/T 17975.9—2000 信息技术 运动图像及其伴音信息的通用编码 第 9 部分 系统解码器的实时接口扩展

GB/T 16649.1—1996 识别卡 带触点的集成电路卡 第一部分：物理特性

GB/T 16649.2—1996 识别卡 带触点的集成电路卡 第二部分：触点的尺寸和位置

GB/T 16649.3—1996 识别卡 带触点的集成电路卡 第三部分：电信号和传输协议

GB/T 14219—1993 中文图文电视广播规范

GY/Z 174—2001 数字电视广播业务信息规范

PC 卡标准，第 2 卷 电气特性规范，1995 年 2 月，PCMCIA 国际协会

PC 卡标准，第 3 卷 物理特性规范，1995 年 2 月，PCMCIA 国际协会

PC 卡标准，第 4 卷 Metaformat 规范，1995 年 2 月，PCMCIA 国际协会

3 定义

3.1 **传送流** transport stream

在 GB/T 17975.1 中定义的一种系统流数据结构。

3.2 **多密** multicrypt

指接收机中对多个不同的条件接收系统的节目进行接收的技术或方式。

3.3 **复用** multiplex

在一个物理频道中承载一个或多个业务或事件的所有数据流。

3.4 **复用器** multiplexer（MUX）

国家广播电影电视总局 2001-09-18 批准

对数据流进行复用的设备。

3.5 **公共接口** common interface

多密技术中主机和模块之间的接口，包括物理层和逻辑层。

3.6 **广播者（业务提供商）** broadcaster（service provider）

一个按照一定计划组织一系列事件或业务并向用户发送的组织或机构。

3.7 **加密/解密系统** encryption/decryption system

在条件接收系统中存在两种类型的加密，用途如下：

— 对授权管理信息 EMM 进行加密处理，然后以单独授权或分组授权的方式发送到用户接收终端的相应处理装置；

— 对授权控制信息 ECM 进行加密处理，其中 ECM 信息中包含了对业务的访问准则信息以及用于解扰的信息。

解密操作在接收机端进行。通常为了安全性，解密操作和接收机分离，在一个可分离的模块中进行，如智能卡，以利于增强系统的保密性。

3.8 **加扰/解扰系统** scramble/descramble system

加扰是为了保证传输安全而对业务码流进行特殊处理。通常在广播前端的条件接收系统控制下改变或控制被传送业务码流的某些特性，使得未经授权的接收者不能得到正确的业务码流。

解扰是加扰的逆过程，在用户接收端的解扰器中完成。

3.9 **加扰周期** crypto period（CP）

加扰器使用某一个控制字的时间段长度。

3.10 **禁用** forbidden

表示某个值不应被使用。

3.11 **客户端** client

使用服务器主机端的一项或多项资源的软件。

3.12 **控制字** control word（CW）

用于加扰的控制信息。

3.13 **控制字发生器** control word generator（CWG）

用于产生控制字的部件。

3.14 **模块** module

多密技术中，指不能独立工作的部件，必须和主机相连，根据应用（如 CA、EPG）的需要来运行特定的任务。这些应用并不直接由主机提供。

3.15 **授权管理信息** entitlement management message（EMM）

一种专有的条件接收信息，指定了用户或用户组对业务或事件的授权等级。

3.16 **授权管理信息发生器** entitlement management message generator（EMMG）

产生授权管理信息的部件，并按照合适的时间间隔重发。

3.17 **授权控制信息** entitlement control message（ECM）

一种专有的条件接收信息，包含有经安全加密的控制信息和授权信息。

3.18 **授权控制信息发生器** entitlement control message generator（ECMG）

产生授权控制信息的部件。

3.19 **通道** channel

代表 TCP 连接的一个应用，并允许应用来决定该连接的参数。通常对应一对一的 TCP 连接。

3.20 **同密** simulcrypt

指通过同一种加扰算法和加扰控制信息，使多个条件接收系统一同工作的技术或方式。

3.21 **同密同步器** simulcrypt synchroniser (SCS)

一种获得控制字和授权控制信息，并使所有与之相连的条件接收系统同步播出的逻辑部件。

3.22 **业务** service

作为节目提供给用户的一套基本流。它们通常由不同数据组成，如视频、音频、字幕、其他数据等，通过公共同步关联起来。

3.23 **业务信息** service information (SI)

在传送流中传送的一种信息，可帮助浏览和事件选择。

3.24 **应用** application

多密技术中，指在模块上运行的和主机进行通信的程序，为使用者提供主机所能提供的功能，一般应可以处理传送流。

3.25 **智能卡** Smart Card、IC 卡 Integrated Circuit Card

在本指导性技术文件中，“IC 卡”与“智能卡”含义相同，均指用于接收机中条件接收系统模块的集成电路卡。

3.26 **主机** host

指由 IP 地址唯一确定的计算机系统、以及计算机网络中可唯一设定 IP 地址的计算机系统。它也可能同时承担客户端和服务器的作用。

多密技术中，特指用户接收设备，如 IRD、VCR、PC 等，可以与模块相连。

3.27 **主机前端** host head-end

前端系统中除了条件接收之外的其他部分。

3.28 **专有** proprietary

表示某个接口由前端供应商或 CA 系统供应商规定。这个接口可以商业公开，但在本指导性技术文件中不公开，它可以通过签定商业或技术许可协议得到。

3.29 **转移控制** trans-control

指在不改变加扰信号，不解扰，也不再次加扰的情况下，改变条件接收系统的过程。

3.30 **资源** resource

指服务器上可提供的一整套功能。在单个主机上可以驻有一个或多个资源。

多密技术中，特指主机为模块提供的功能单元。它定义了一整套在模块和主机之间交换的对象、模块借此使用资源。

4 缩略语

AF	Adaptation Field	适配域
APDU	Application Protocol Data Unit	应用协议数据单元

ASN	Abstract Syntax Notation	抽象语法符号
Bslbf	bit string, left bit first	比特串，左位在先
CA	Conditional Access	条件接收
CAM	Conditional Access Module	条件接收功能模块
CAS	Conditional Access System	条件接收系统
CAT	Conditional Access Table	条件接收表
CI	Common Interface	公共接口
CP	Crypto Period	加扰周期
CPU	Centre Process Unit	中央处理器
CRC	Cyclic Redundancy Check	循环冗余校验
CW	Control Word	控制字
CWG	Control Word Generator	控制字发生器
ECM	Entitlement Control Message	授权控制信息
ECMG	Entitlement Control Message Generator	授权控制信息发生器
EIS	Event Info Scheduler	事件信息调度器
EIT	Event Information Table	事件信息表
EMM	Entitlement Management Message	授权管理信息
EMMG	Entitlement Management Message Generator	授权管理信息发生器
EPG	Electronic Program Guide	电子节目指南
ID	Identifier	标识符
IP	Internet Protocol	因特网协议
IPPV	Impulse Pay Per View	即时按次付费节目
IRD	Integrated Receiver Decoder	综合接收解码器
ISO	International Organization for Standardization	国际标准化组织
LSB	Least Significated Bit	最低有效位
MJD	Modified Julian Date	修正的儒略日期
MMI	Man Machine Interface	人机界面
MPEG	Moving Pictures Expert Group	运动图像专家组
MSB	Most Significant Bit	最高有效位
MUX	Multiplexer	复用器
NDA	Non Disclosure Agreement	保密协议
NIT	Network Information Table	网络信息表
OSD	On Screen Display	屏幕显示
OSI	Open Systems Interconnect	开放系统互连
PAT	Program Association Table	节目关联表
PCMCIA	Personal Computer Memory Card International Association	个人计算机存储卡国际协会
PDG	Private Data Generator	专用数据发生器
PES	Packet Elementary Stream	打包基本流
PID	Packet Identifier	包标识符

PMT	Program Map Table	节目映射表
PPC	Pay Per Channel	按频道付费
PPV	Pay Per View	按次付费
PSI	Program Specific Information	节目特定信息
rpchof	Remainder Polynomial Coefficients，Highest Order First	多项式余数，高阶在先
SAS	Subscriber Authorization System	用户授权系统
SC	Smart Card	智能卡
SCR	Scrambler	加扰器
SCS	Simulcrypt Synchroniser	同密同步器
SDT	Service Description Table	业务描述表
SI	Service Information	业务信息
SIG	Service Information Generator	业务信息发生器
SMS	Subscriber Management System	用户管理系统
SPDU	Session Protocol Data Unit	会话协议数据单元
ST	Stuffing Table	填充表
STB	Set Top Box	机顶盒
tcimsbf	two’s complement integer msb (sign) bit first	2 的补码，高位(符号位)在先
TCP	Transport Control Protocol	传输控制协议
TDT	Time and Date Table	时间和日期表
TLV	Type，Length，Value	类型，长度，值
TPDU	Transport Protocol Data Unit	传输协议数据单元
TS	Transport Stream	传送流
UDP	User Datagram Protocol	用户数据报协议
uimsbf	unsigned integer most significant bit first	无符号整数，高位在先
UTC	Universal Time Coordinated	世界协调时

5 条件接收系统描述

条件接收是指这样一种技术手段，它只容许被授权的用户使用某一业务，未经授权的用户不能使用这一业务。条件接收系统正是实现该功能的系统。

条件接收系统能实现各项数字电视广播业务的授权管理和接收控制。该系统是一个综合性的系统，系统涉及了多种技术，包括系统管理技术、网络技术、加解扰技术、加解密技术、数字编解码技术、数字复用技术、接收技术、智能卡技术等，同时也涉及到用户管理、节目管理、收费管理等信息管理应用技术。

条件接收系统是数字电视接收控制的核心技术保障系统。该系统可以按不同情况对数字电视广播业务按时间、频道和节目进行管理控制。

在用户端，未经授权的用户将不能对加扰节目进行解扰，而无法收看该节目。条件接收是现代信息加密技术在数字电视领域的具体应用，一方面实现了节目及业务信息的分类和管理；另一方面可实现节目的条件接收。

为使接收机能够在不同的 CA 系统情况下对节目进行解扰，可以采用本指导性技术文件附录中规

定的两种技术：

—同密；

—多密。

在某些情况下，可采用转移控制，如：从卫星、地面传输网接收到的加扰节目需要进入到有线电视网络，有线网络运营商通过转移控制，可使本网络中的原有接收机接收。

6 条件接收系统的构成

典型的条件接收系统由用户管理系统、节目信息管理系统、加密/解密系统、加扰/解扰系统等构成，其逻辑结构如图 1 所示。

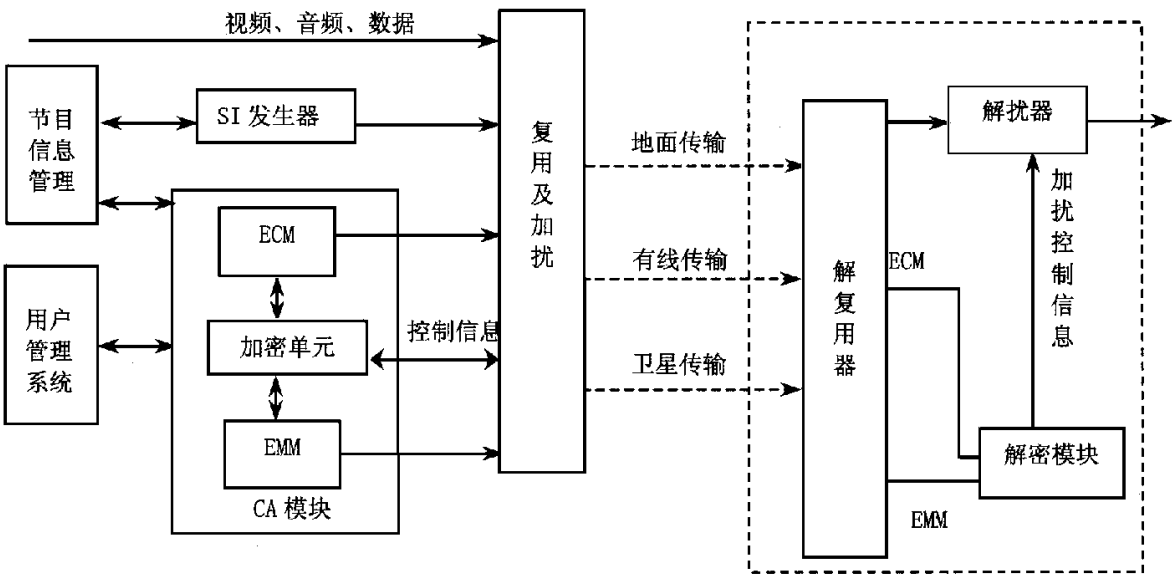


图 1 典型的条件接收系统逻辑结构

其中系统各部件之间通过相关接口进行通信和数据传输，主要包括以下接口：节目信息管理接口、用户管理系统接口、复用器接口、智能卡接口等。

6.1 用户管理系统

用户管理系统主要实现数字电视广播条件接收用户的管理，包括对用户信息、用户设备信息、用户预订信息、用户授权信息、财务信息等进行记录、处理、维护和管理。

6.1.1 系统管理

系统管理负责对整个用户管理系统进行初始化设置和参数配置以及其他系统管理工作，完成系统及子系统之间的配置、管理、控制和执行，并定义系统与其他系统的接口。系统管理的主要目的是保证系统能安全可靠地运行。

6.1.2 用户管理

主要功能：编辑和管理用户信息，处理用户的节目定单，检查用户付费情况，产生用户的预授权信息。

6.1.3 用户授权管理

用户授权管理负责用户业务开通前的授权预处理操作，主要包括对用户信用度的确认、用户业务与智能卡有效性确认等。

6.2 节目信息管理系统

节目管理为即将播出的节目建立节目表。节目表包括频道，日期和时间安排，也包括要播出的各个节目的CA信息。

节目管理信息被SI发生器用来生成SI/PSI信息，被播控系统用来控制节目的播出，被CA系统用来做加扰调度和产生ECM，同时送入SMS系统。

6.3 加扰/解扰系统

加扰是为了保证传输安全而对业务码流进行特殊处理。通常在广播前端的条件接收系统控制下改变或控制被传送业务码流的某些特性，使得未经授权的接收者不能得到正确的业务码流。

解扰是加扰的逆过程，在用户接收端的解扰器中完成。

6.4 加密/解密系统

在条件接收系统中存在两种类型的加密单元，用途如下：

—对授权管理信息EMM进行加密处理，然后以单独授权或分组授权的方式发送到用户接收终端的相应处理装置；

—对授权控制信息ECM进行加密处理，其中ECM信息中包含了对业务的访问准则信息以及用于解扰的信息。

解密操作在接收机端进行。通常为了安全性，解密操作和接收机分离，在一个可分离的模块中进行，如智能卡，以利于增强系统的保密性。

7 条件接收系统的总体要求

7.1 全国条件接收系统标识

为了便于对全国条件接收系统进行统一规划管理，对全国条件接收系统的标识号定义见表1。

表1 标识取值范围

	主标识（2字节）	辅标识（2字节）
取值范围	0x0000——0xFFFF	0x0000——0xFFFF
注：高位在前		

主标识用于区分不同条件接收系统供应商的系统，辅标识由条件接收系统供应商自定义。

7.2 基本要求

7.2.1 基本功能

条件接收系统能够控制用户接收包括视频、音频和数据在内的数字广播服务。

7.2.2 传输方式

条件接收系统能支持包括卫星、有线和地面等网络在内的多种传输方式。

7.2.3 语言

条件接收系统应以中文语言显示相关系统信息和文本信息。

7.2.4 数据广播

条件接收系统能完全地支持音、视频以外的增值数据服务。

7.2.5 交互应用

条件接收系统能支持交互应用以及相应的服务授权和控制。

7.3 节目信息管理要求

系统容量：条件接收系统能够支持足够多的频道或业务。

7.4 用户管理要求

容量可扩展性：条件接收系统能够支持足够多的用户，并可支持升级扩充。

灵活性：一个条件接收系统可以同时有多个用户管理系统，或者一个用户管理系统同时为多个条件接收系统共享。

寻址能力：条件接收系统具备对用户进行授权或取消授权的功能，以控制用户对视频，音频，数据及运营商相关信息的接收，授权方式为：单独用户、组用户。

系统支持以下几种分组方式：

—用户以地理位置分组，如邮政编码；

—用户由所选的运营商的某些统计或市场信息进行逻辑分组；

—用户以其共享的硬件状况或数据特征分组，如按照条件接收模块（智能卡）、接收机通用模块或软件版本信息进行分组。

区域限播：条件接收系统提供阻止某些地理区域对特定节目的收视功能，如以地理区域的邮政编码划分。

家长控制：条件接收系统能对播放的节目进行等级划分，并为用户提供可控的收视口令。

7.5 应急广播要求

应满足与条件接收系统有关的应急广播要求。

7.6 智能卡要求

采用智能卡的接收机并不是唯一的形式，如果接收机采用智能卡，应满足以下要求：

— 条件接收系统使用符合 GB/T 16649 的智能卡；

— 条件接收系统能够提供智能卡与接收机之间的相互认证及配对，以保证智能卡使用其指定的接收机；

— 全国范围内应使用统一的标识。

智能卡的基本的安全要求如下：

物理保密：若不凭借特别的技巧和工具，无法对软件和硬件作增删或替换；只有通过特殊手段，才能对敏感数据进行输入、存储、访问与修改；任何部分的故障或破坏均不会导致敏感数据泄漏。

逻辑保密：当处于一种敏感状态时，不许有两个以上的委托方的参与；口令的输入必须以保护其他敏感数据的相同方式得到保护。

密钥管理：密钥算法必须采用国家已经批准的算法，并且严格保管密钥。

7.7 智能卡与接收机通信和安全认证

智能卡和接收机之间需要进行相互合法性认证，该认证一般是由条件接收系统供应商自己指定的算法完成；智能卡与接收机间的通信需进行加密，以达到强化商业安全的目的。

7.8 系统安全性要求

—可变化的加扰（控制信息变化周期）：条件接收系统能提供变化的加扰控制信息，其变化的周期一般为5秒～30秒；

—认证：除了数字电视广播系统和条件接收系统之间的安全外，条件接收系统在接收机和智能卡间的信息交换应考虑采用双向认证的原则，以利于增强对未经认证的侵入、盗版接收机的销售及智能卡非法复制的控制。

条件接收系统的安全认证包括对系统管理员的身份认证、对节目提供者的身份认证，还包括用户和智能卡的认证、接收机和智能卡之间的认证。

7.9 回传信道的需求

建立回传信道：条件接收系统可以建立与接收机的回传信道。

7.10 同密与多密

为解决多个条件接收系统的相互兼容，可以参照附录的有关章节采用同密或多密的解决方式。

7.11 管理体系和发卡体系

为了保障整个条件接收系统的安全性，需对密钥管理体系和发卡体系进行管理。

附 录 A
（文件的附录）
节目信息管理与条件接收系统接口和要求

A1 节目管理信息

节目管理为即将播出的节目建立节目表。节目表包括频道，日期和时间安排，也包括要播出的各个节目的CA有关信息。

节目管理信息被SI发生器用来生成SI/PSI信息，被播控系统用来控制节目的播出，被CA系统用来做加扰调度和产生ECM，同时送入SMS系统。

除下面描述的一些信息以外，在一个关键的基本系统中ECM实际上要包括用于解扰数据包的CW。

在节目管理信息中针对CA业务所定义的描述符为强制性的。

A2 CA业务描述符

此描述符定义了与CA业务有关的总信息。有关业务更详细的信息在紧跟于此描述符之后的独立描述符中有所涉及。

A2.1 CA业务ID描述符

此描述符为前面定义的CA业务定义了一个ID。CA业务ID描述符具备以下信息：CA业务ID（由CAM使用）。

A2.2 起始时间/终止时间描述符

此描述符具有UTC.MJD格式的时间信息。

A2.3 价格描述符

价格描述符具有以下信息：“值”相当于“节目的价格”。

A2.4 家长控制级别描述符

家长控制有多个级别。

A2.5 配对描述符

配对描述符具有以下信息：“配对”相当于“配对关或开”。

A2.6 地理寻址描述符

此描述符定义了一个地址，包含了一个已定义的地理参考。

A2.7 加扰模式描述符

定义一个节目是否被加扰。

A2.8 接收准则描述符

节目对应的接收准则信息。

A3 接口要求

A3.1 通信协议

支持以太网协议。

A3.2 其他要求

节目管理要能够同时支持多个CA系统。

附 录 B
（文件的附录）
用户管理系统的要求及其接口

B1 概述

用户管理系统主要是实现数字电视广播条件接收用户的管理，包括对用户信息、用户设备信息、用户预订信息、用户授权信息、财务信息等进行记录、处理、维护和管理。

用户管理系统的界面必须使用中文。

B2 基本功能

B2.1 用户信息管理

用户信息管理负责完成的主要功能是处理与维护用户的主要信息。包括：

—用户基本信息，诸如用户号、用户姓名、证件类型与号码等；

—用户信用等级信息；

—用户合同信息。

具体功能包括：

—用户信息的注册；

—用户信息的修改；

—用户信息的删除。

B2.2 资源管理

资源管理的内容是业务服务所需设备产品，如智能卡、接收机等，主要功能是对资源进行进销调存以及使用状态的管理，包括用户已经购买或租赁使用的资源，以及对资源信息进行管理。

B2.3 产品管理

产品管理主要是对节目提供商所提供的节目进行产品化的管理。管理的内容包括业务服务在系统中的注册、修改与删除，业务服务的促销、定价和有效期确认等。业务产品信息主要包括：

—产品代码；

—产品说明；

—产品定价；

—产品类型；

—产品有效时间。

产品管理具体功能包括：

—产品信息生成；

—产品信息修改；

—产品信息删除。

B2.4 用户业务管理

用户业务管理负责用户订购产品信息的管理。其管理具体功能包括：

—用户订购业务信息的生成；

—用户订购业务信息的修改；

—用户订购业务信息的取消。

B2.5 用户业务支援

业务支援管理的内容是为保证用户业务正常运行而进行的管理功能，主要包括用户业务故障单、故障维修或解决的结果等。

B2.6 计费

计费主要是计算应收取用户的费用。支持的计费方式有：按频道计费、按时间计费、按次计费等，包括实现对 PPV 计费的功能。

B2.7 收费

系统在计费处理之后，进行收费。

B2.8 账务管理

账务管理的主要功能包括对应收和实收的关系进行管理、对用户信用进行管理、对财务清算和账务报表的管理等。

B2.9 授权管理

授权管理负责用户业务开通前的授权预处理操作，主要包括对用户信用度的确认、用户业务与智能卡有效性确认，以及通过用户管理系统接口向条件接收系统的授权模块发送授权请求等功能。

积极采用合理的算法或数据交换结构，提高与条件接收系统授权模块的数据交换速度。

B2.10 报表管理

按照不同的条件与参数，对基于用户管理系统的数据进行运营情况的统计分析，生成各类分析报表，向管理者及运营商提供各种分析数据。

B2.11 系统管理

为保证系统正常运行所需要的系统附加功能，具体的包括：

—参数管理；

—系统和数据的备份管理。

B2.12 区域管理

根据用户的地理位置，按管理要求对用户进行分级别、分区块地设置与管理。

B3 接口规范

此接口的描述是基于假想的系统进行调用的形式。这些调用并没有描述必需的代码和实现方式，只是陈述了其支持的功能。

B3.1 物理接口

采用以太网协议。

B3.2 网络协议

SMS-CAS 接口采用 TCP/IP 协议与用户管理系统（SMS）进行通信，面向 SMS 系统的一端采用 TCP 协议。

B3.3 针对用户的操作

B3.3.1 发送用户资料

SMS 通过此命令与 CAS 进行初始交互，向 CAS 提供用户的详细资料以便 CAS 可以针对特定的用户生成 EMM 消息。

这个命令给 CAM 发送了可以使用的授权，但是这个 CAM 并没有取得任何服务的授权。

B3.3.2 删除用户

此命令从 SMS 中分离一个用户，取消这个 SMS 提供给此用户的所有服务。在单一 SMS 系统中，取消此用户的智能卡。

B3.3.3 创建/更新寻址数据

定义与用户位置相关联的地理区域，用于寻址或实现 CA 功能，如：区域禁播。

B3.3.4 修改用户资料

此命令允许 SMS 更新或修改用户资料，例如：返回报告消息设置，寻址数据等。

B3.3.5 挂起用户/重新激活用户

此命令允许 SMS 暂时挂起一张智能卡已有的所有服务，过一段时间后再重新激活。不需要逐个地取消用户的服务然后再重新分配给用户。

B3.3.6 添加/取消/更新授权

服务可以是连续的服务即可更新的服务，这种服务基于一定的规则接受主动的授权，也可以是一个在某个特定时间到期的预定的 PPV 服务。

B3.3.7 取消所有服务

此命令并没有注销用户的智能卡，只是终止了这张智能卡的所有服务授权。

B3.3.8 发送邮件

向接收机发送邮件。

B3.3.9 发送屏显消息

发送给接收机，在一段时间内直接显示在屏幕上的短消息。

B3.3.10 清除密码

此命令允许 SMS 发送一条消息清除掉接收机用户设置的密码，把它设置回默认值。在用户忘记密码的情况下使用。

B3.3.11 配对/取消配对

配对是指将用户的智能卡和某一个特定的接收机关联起来。某些广播事件可能只能要求与接收机进行初始配对的那张卡。这条命令可以取消原始的配对信息，令当前使用的卡与接收机重新配对。

附 录 C
（文件的附录）
智能卡及其与接收机接口的要求

C1 智能卡物理接口

采用 GB/T 16649 接口规范。

C2 智能卡通信加密机制

智能卡通信加密提供第二层的数据保护，以防止智能卡中的数据被篡改或攻击。通过配对过程中创建加扰器密钥，提供了一机一卡特性。

C3 加密处理器

智能卡的体系结构可以选择是否支持多个加密算法，通过 CA 系统的注册指令可以将被选的加密处理器打开。加密算法可由 CA 系统确定。

C4 密码和个人信息

存放有区域密码和用户信息的智能卡存储保护区。

C5 智能卡指令

C5.1 智能卡和接收机配对/解除配对

智能卡必须支持配对和解除配对。

C5.2 系统注册

系统初始化时，CA 系统内核应该通过 CA 系统向智能卡发出注册请求指令，并根据只读信息文件中的数据来选择加载默认加密程序。

C5.3 只读 CA 信息

用于访问只读 CA 信息文件。

C5.4 CA 数据文件访问

用于访问受密码保护的 CA 数据文件。

C5.5 用户信息访问

用于访问用户信息文件。

C5.6 预订记录处理

用于从预订记录文件中读取、添加和删除预订记录。一个存储在智能卡中的程序将检查事件/服务的权限。

C5.7 对话处理

用于增加或减少在智能卡中对话文件的标记值。

C5.8 交易记录处理

用于创建、删除、读取智能卡中交易记录文件所包含的交易记录。

C5.9 取得智能卡版本信息

见表 C1。

表 C1 智能卡版本信息

属性	说明
CAS _ ID	CA 系统的标识符
Version	CA 系统的版本号
Desc	CA 系统的说明

C5.10 取得用户信息

见表 C2。

表 C2 用户信息

属性	说明
CARD _ NO	卡的编号
Date	发行日期
Expire _ date	过期日期
Provider	发行商说明

附 录 D
（文件的附录）
条件接收智能卡标识

＃＊2〗NO）反映执卡用户的消费属性。智能卡 ID 号、智能卡编号在全国具有唯一性，由全国统一发号。

从实用性、可扩展性、安全性等几个方面对智能卡地址号（CARD _ ID）、智能卡编号（CARD _ NO）的定义如下。

D1 智能卡地址号（CARD _ ID）

智能卡地址号是对智能卡授权管理时寻址使用的唯一标识，必须全国统一编码。

智能卡地址号由两部分构成，见表 D1。

表 D1 智能卡地址号

第 1 字节 预留 （高位）	第 2 字节 预留	……
第一部分		第二部分

第一部分是预留的。

第二部分至少 4 个字节。

D2 智能卡编号（CARD _ NO）

智能卡编号（CARD _ NO），每个编号有 16 个 10 进制数，见表 D2。

表 D2 智能卡编号

1 （高位）	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

智能卡编号必须全国统一编码。

CARD _ ID 与 CARD _ NO 之间有对应的关系。

附 录 E
（文件的附录）
同密技术

E1 概述

本附录规定了同一个前端、两个或多个条件接收系统互操作的所有要求，包括系统架构、定时关系、消息结构。

系统架构内的部件代表功能模块。实际上物理器件的边界不需要和功能模块的边界完全吻合。例如 SCS 可能在 MUX 之中，也有可能 SCS 和 MUX 单独分开，两种结构都可行。

考虑到同密的架构，要求所有的同密系统基于共享的加扰和解扰算法。

E2 系统架构

图 E1 给出了各部件间的逻辑关系，并在本附录中规定了部件之间的接口。前端系统中还有一些本图中未表示的部件，如 SMS。

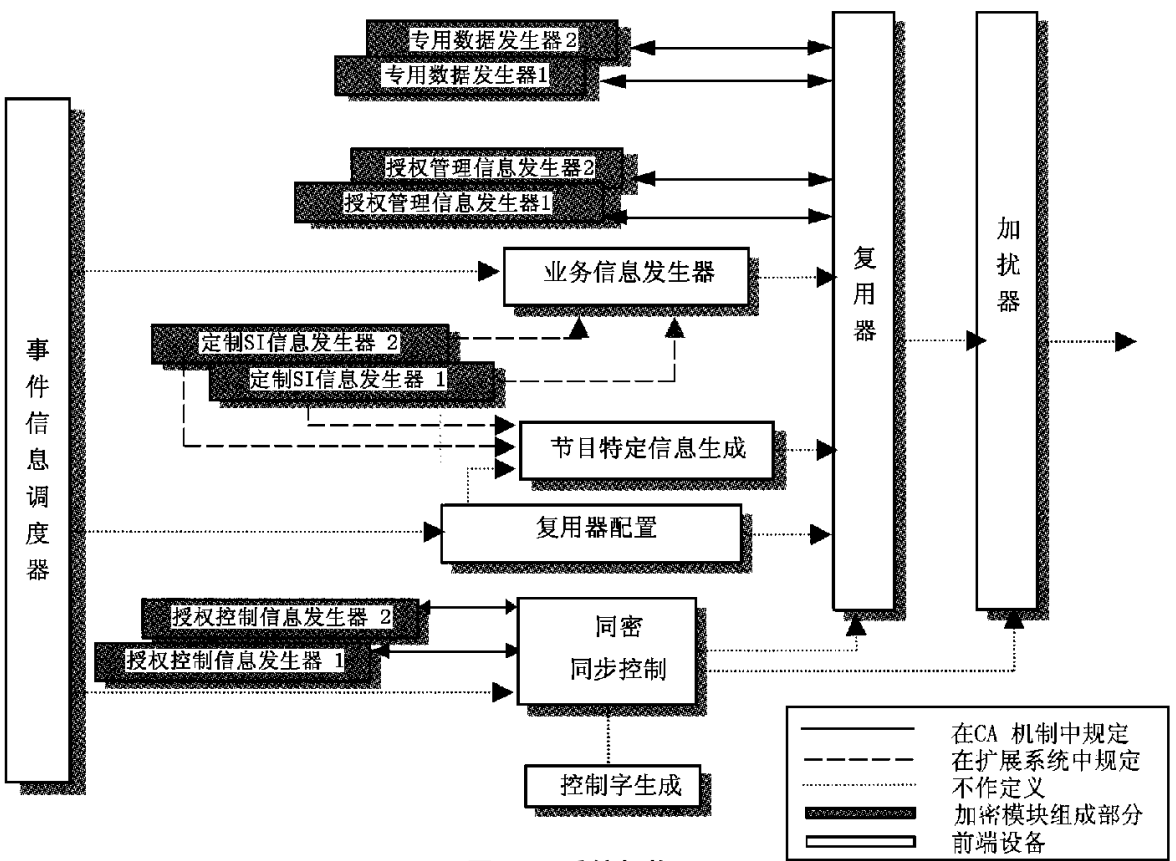


图 E1 系统架构

上面的同密系统架构分为两个部分：

E2.1 前端部件

指在同密前端系统中，除了同密部件之外的其他部件。

E2.2 同密 CA 部件

典型情况是指一个新的 CA 系统供应商加入到同密前端系统中的部件。必须指明的是，在同密系统中不一定必须有 EMMG、PDG、用户 SI 发生器。

E3 部件描述

E3.1 事件信息调度器（EIS）

在同密系统架构中，EIS 功能模块负责处理整个系统所有的时间计划信息、配置、CA 特定信息。EIS 是前端系统中的存储数据库。例如，它负责通过 SCS 为 ECMG 提供它们所需要的信息来产生 ECM。实际中，EIS 可以分布在不同的物理单元、存储位置、输入终端中，也可以和其他任何功能模块进行通信。

CA 系统供应商的部件和 EIS 的连接、传输的数据必须通过广播者的确认，在本附录中未对此作出规定。

E3.2 同密同步控制（SCS）

同密同步控制的作用是：

—和 ECMG 建立 TCP 连接，为每个连接建立一个通道；

—在通道内建立所需要的流，并分配 ECM _ stream _ ID 值；

—从 CWG 取得控制字 CW；

—通过相关的流向 ECMG 提供控制字和相关 CA 信息；

—从 ECMG 取得 ECM；

—根据和频道参数相关的相应加扰周期实现同步；

—把 ECM 传送给复用器，并要求它们按照频道参数重复；

—提供控制字 CW 给加扰器，以在特定的加扰周期内使用。

E3.3 授权控制信息发生器（ECMG）

ECMG 接收 CW 标准信息中的控制字 CW，并返回 ECM 或者错误信息。ECMG 并不支持 ECM 的重复。

E3.4 授权管理信息发生器（EMMG）

这个由 CA 系统供应商提供的部件通过同密接口和 MUX 接口。由 EMMG 初始化到 MUX 的连接。

E3.5 专用数据发生器（PDG）

在同密系统结构图中使用这个功能模块是为了表明，EMMG 和 MUX 的接口可以传送 EMM 和其他 CA 相关的私有信息。由 PDG 初始化和 MUX 的连接。

E3.6 定制 SI 发生器（CSIG）

这个部件负责产生专用 SI 信息。它和 SI、PSI 发生器接口。这些接口在本附录中未规定。

E3.7 复用器配置（Mux Config）

这个部件负责配置 MUX 并且提供到 PSI 发生器的连接，以便产生 PSI 和播出。在本附录中未规定 MUX 配置和 MUX、MUX 配置和 PSI 发生器接口。

E3.8 SI 发生器（SIG）

这个部件负责产生系统的 SI 信息。SI 服务器从 EIS 取得基本数据，从用户 SI 服务器取得由 CA 系统供应商提供的附加信息。EIS 和 SI 服务器、SI 服务器和 MUX 的接口在本附录中未规定。

E3.9 PSI 发生器（PSIG）

这个部件负责产生系统的 PSI 信息。PSI 服务器从 MUX 配置取得系统的基本数据，从用户 SI 服务器取得由 CA 系统供应商提供的附加信息。

E3.10 复用器（MUX）

这个前端部件的作用是对输入数据进行复接，输出 MPEG-2 TS 流。输入可以是 TS 包、MPEG-2 段、原始数据。MUX 的准确功能实际上是个工具，可以和 SCS 通信，可以和 EMMG 建立连接。

E3.11 加扰器（SCR）

在本附录中未规定相关接口。

E3.12 控制字发生器（CWG）

在本附录中未规定相关接口。

E4 接口描述

E4.1 ECMG 与 SCS 的接口

与通道有关的消息

该接口定义了以下与通道有关的消息：

—Channel _ setup；

—Channel _ test；

—Channel _ status；

—Channel _ close；

—Channel _ error。

流相关消息

该接口定义了以下与通道有关的消息：

—Stream _ setup；

—Stream _ test；

—Stream _ status；

—Stream _ close _ request；

—Stream _ close _ response；

—Stream _ error；

—CW _ provision；

—ECM _ response。

接口原则

对接口来说，SCS 是客户端、ECMG 是服务器。SCS 事先知道 ECMG 的 Super _ CAS _ ID、IP 地址、端口号之间的映射关系。当 EIS 请求一个给定 Super _ CAS _ ID 值的 ECM 流时，SCS 将打开与恰当的 ECMG 之间的流。这或许会需要打开一个新的通道（包括打开一个新的 TCP 连接）。

注：对同一个 Super _ CAS _ ID 值，可能会有多个 ECMG（例如为了性能和冗余）。这种情况下，SCS 将根据冗余策略或资源情况选择一个 ECMG 建立连接。

通道建立

对一个 TCP 连接来说有且只有一个通道。一旦 TCP 连接建立，SCS 发送 Channel _ setup 消息给

ECMG。如果成功，ECMG 返回一个 Channel _ status 消息。

如果拒绝或失败，ECMG 返回 Channel _ error 消息，意味着 ECMG 未打开通道，SCS 将关闭 TCP 连接。

流建立

SCS 发送 Stream _ setup 消息给 ECMG。如果成功，ECMG 返回 Stream _ status 消息。如果拒绝或失败，ECMG 返回 Stream _ error 消息。一旦连接建立、通道和流正确建立，ECM 将被传送。ECM 可以用段或 TS 包的方式传送。ECMG 在通道建立时表明将使用那种数据对象。一旦连接建立，通道和流正确建立，ECM 将作为对 CW _ provision 消息的响应被传送到 SCS。

流关闭

当 ECM 流不再需要或者发生错误时会被关闭，ECM 流关闭总是由 SCS 发起。它是通过 Stream _ close _ request消息实现的。Stream _ close _ response 消息指示流已被关闭。

通道关闭

通道关闭在通道不再存在或发生错误时发生（由 SCS 检测到或 ECMG 报告）。它通过 SCS 发送 Channel _ close 消息实现。作为结果，连接的两端都被关闭。

通道/流状态测试和状态

任何时刻两个部件之一都可以发送 Channel _ test/Stream _ test 消息来检查通道/流的完整性。作为对此消息的应答，接收部件发送通道/流状态消息或通道/流错误消息。

未预见的通信丢码

SCS 和 ECMG 都可以处理通信丢码问题（连接、通道或流中的丢码）。每个部件怀疑可能的通信丢码（例如 10 秒之内无数据），可以发送测试消息来检查通信状态，并期望收到一个状态消息。如果在给定时间内未收到状态消息（与具体实现有关），将重新建立通信链路。

处理数据的不一致性

如果 ECMG 收到不一致的数据，它将向 SCS 发送错误消息。如果 SCS 收到这样的消息或检测到不一致性，它将关闭连接。SCS（作为客户端）将建立或重新建立连接、通道或流。

注：当发送用户定义的数据、未知的参数类型或消息类型，不应被认为是不一致性。

E4.2 EMMG 与 MUX 的接口

通道相关消息

这个接口相关的消息有：

—Channel _ setup；

—Channel _ test；

—Channel _ status；

—Channel _ close；

—Channel _ error。

流相关消息

这个接口相关的消息有：

—Stream _ setup；

—Stream _ test；

—Stream _ status；

—Stream _ close _ request；

—Stream _ close _ response；

—Stream _ error；

—Data _ provision。

接口原则

对这个接口，EMMG 是客户端，MUX 是服务器端。

通道建立

EMMG 发送 Channel _ setup 消息给 MUX。如果成功，MUX 发回一个 Channel _ status 消息。

如果拒绝或失败，MUX 返回 Channel _ error 消息，意味着通道未被 MUX 打开，EMMG 将关闭 TCP 连接。

流建立

EMMG 发送 Stream _ setup 消息给复用器。如果成功，MUX 返回 Stream _ status 消息。如果失败，MUX 返回 Stream _ error 消息。一旦连接建立、通道和流正确建立，EMM 将被传送。EMM 可以用段或 TS 包传送。ECMG 在通道建立时指出将使用那种数据对象。

带宽分配

这个接口允许在 EMMG 和 MUX 之间协商所使用的带宽，这不是强制性的要求。在流建立时，EMMG 将请求那个流的最佳带宽，MUX 返回为该流分配的带宽。EMMG 以后还可以请求对带宽做出调整。MUX 也可以在没有 EMMG 请求的情况下重新分配带宽。

流关闭

当 EMM 流不再需要时会被关闭，EMM 流关闭总是由 EMMG 发起。它是通过 Stream _ close _ request 消息实现的。Stream _ close _ response 消息指示流已被关闭。

通道关闭

通道关闭在通道不再存在或发生错误时产生（由 EMMG 检测到或 MUX 报告）。它通过 EMMG 发送 Channel _ close 消息实现。作为结果，连接的两端都被关闭。

通道/流状态测试和状态

任何时刻两个部件之一都可以发送 Channel _ test/Stream _ test 消息来检查通道/流的完整性。作为对此消息的应答，接收部件发送通道/流状态消息或通道/流错误消息。

未预见的通信丢码

MUX 和 EMMG 都可以处理通信丢码问题（连接、通道或流中的丢码）。每个部件怀疑可能的通信丢码（例如 10 秒之内无数据），可以发送测试消息来检查通信状态，并预期收到一个状态消息。如果在给定时间内未收到状态消息（与具体实现有关），将重新建立通信链路。

处理数据不一致性

如果 MUX 检测到不一致的数据，它将向 EMMG 发送错误消息。如果 EMMG 收到这样的消息或检测到不一致性，它将关闭连接。EMMG（作为客户端）将建立或重新建立连接、通道或流。

注：当发送用户定义的数据、未知的参数类型或消息类型，不应被认为是不一致性。

E4.3 PDG 与 MUX 的接口

这个接口定义同 EMMG 与 MUX 的接口。

E4.4 定制 SI 发生器与 PSI 发生器的接口

在本附录中未定义。

E4.5 定制 SI 发生器与 SI 发生器的接口

在本附录中未定义。

E4.6 EIS 与 SI 发生器的接口

在本附录中未定义。

E4.7 SI 发生器与 MUX 的接口

在本附录中未定义。

E4.8 EIS 与复用配置的接口

在本附录中未定义。

E4.9 复用配置与 PSI 发生器的接口

在本附录中未定义。

E4.10 PSI 发生器与 MUX 的接口

在本附录中未定义。

E4.11 MUX 与 SCR 的接口

在本附录中未定义。

E4.12 加扰器以后的接口

在本附录中未定义。

E4.13 SCS 与 MUX 的接口

在本附录中未定义。

E4.14 SCS 与 SCR 的接口

在本附录中未定义。

E4.15 SCS 与 CWG 的接口

在本附录中未定义。

E4.16 EIS 与 SCS 的接口

在本附录中未定义。

E5 通用消息描述

E5.1 通用消息结构

对所有接口来说，通常的消息具有以下结构：

```
generic _ message
{
  protocol _ version   1 字节
  message _ type       2 字节
  message _ length     2 字节
  for (i=0; i<n; i++)
   {
```

parameter _ type　　2 字节
parameter _ length　　2 字节
parameter _ value　<parameter _ length>字节
}
}

protocol _ version：标识协议版本的 8 比特域，取值为 0x01。

message _ type：标识消息类型的 16 比特域。

message _ length：标识在 message _ length 之后消息中字节数的 16 比特域。

parameter _ type：标识紧跟其后的参数类型的 16 比特域。在本附录的接口规定部分给出了参数类型的取值表。接收端将忽视不能识别的参数类型，和那个参数类型相关的数据将被丢弃并继续处理剩下的消息。

parameter _ length：标识紧跟其后的 parameter _ value 域的字节数的 16 比特域。

parameter _ value：这个可变长度域给出参数的实际值，其语义和参数类型值类似。

E5.2　消息类型值

见表 E1。

表 E1　消息类型值

消息类型值	相关接口	消息类型
0x0000	预留	预留
0x0001	ECMG⇔SCS	Channel _ set-up
0x0002	ECMG⇔SCS	Channel _ test
0x0003	ECMG⇔SCS	Channel _ status
0x0004	ECMG⇔SCS	Channel _ close
0x0005	ECMG⇔SCS	Channel _ error
0x0006 至 0x0010	预留	预留
0x0011	EMMG⇔MUX	Channel _ set-up
0x0012	EMMG⇔MUX	Channel _ test
0x0013	EMMG⇔MUX	Channel _ status
0x0014	EMMG⇔MUX	Channel _ close
0x0015	EMMG⇔MUX	Channel _ error
0x0016 至 0x0020	预留	预留

表 E1 消息类型值（完）

消息类型值	相关接口	消息类型
0x0101	ECMG⇔SCS	Stream _ set-up
0x0102	ECMG⇔SCS	Stream _ test
0x0103	ECMG⇔SCS	Stream _ status
0x0104	ECMG⇔SCS	Stream _ close _ request
0x0105	ECMG⇔SCS	Stream _ close _ response
0x0106	ECMG⇔SCS	Stream _ error
0x107 至 0x110	预留	预留
0x0111	ECMG⇔MUX	Stream _ set-up
0x0112	ECMG⇔MUX	Stream _ test
0x0113	ECMG⇔MUX	Stream _ status
0x0114	ECMG⇔MUX	Stream _ close _ request
0x0115	ECMG⇔MUX	Stream _ close _ response
0x0116	ECMG⇔MUX	Stream _ error
0x0117	ECMG⇔MUX	Stream _ BW _ request
0x0118	ECMG⇔MUX	Stream _ BW _ allocation
0x0119 to 至 0x0120	预留	预留
0x0201	ECMG⇔SCS	CW _ provision
0x0202	ECMG⇔SCS	ECM _ response
0x0203 至 0x0210	预留	预留
0x0211	ECMG⇔MUX	data _ provision
0x0212 至 0x0220	预留	预留
0x0221 至 0x7FFF	预留	预留
0x8000 至 0xFFFF	用户定义	用户定义

E6 接口特性消息描述

E6.1 ECMG 与 SCS 接口

E6.1.1 Parameter _ type 值

见表 E2。

表 E2 参数类型值

Parameter _ type 值	参数类型	类型/单位	长度（字节）
0x0000	预留	—	—
0x0001	Super _ CAS _ ID	Uimsbf	4
0x0002	section _ TSpkt _ flag	Boolean	1

表 E2 参数类型值（完）

Parameter _ type 值	参数类型	类型/单位	长度（字节）
0x0003	delay _ start	Tcimsbf/ms	2
0x0004	delay _ stop	Tcimsbf/ms	2
0x0005	transition _ delay _ start	Tcimsbf/ms	2
0x0006	transition _ delay _ stop	Tcimsbf/ms	2
0x0007	ECM _ rep _ period	uimsbf/ms	2
0x0008	max _ streams	uimsbf	2
0x0009	min _ CP _ duration	uimsbf/n x100ms	2
0x000A	lead _ CW	uimsbf	1
0x000B	CW _ per _ msg	uimsbf	1
0x000C	max _ comp _ time	uimsbf/ms	2
0x000D	access _ criteria	用户定义	可变的
0x000E	ECM _ channel _ ID	uimsbf	2
0x000F	ECM _ stream _ ID	uimsbf	2
0x0010	nominal _ CP _ duration	uimsbf/n × 100ms	2
0x0011	access _ criteria _ transfer _ mode	Boolean	1
0x0012	CP _ number	Uimsbf	2
0x0013	CP _ duration	uimsbf/n × 100ms	2
0x0014	CP _ CW _ Combination	—	10 (2+8)
	CP	uimsbf	2
	CW	uimsbf	8
0x0015	ECM _ datagram	用户定义	可变的
0x0016 至 0x006F	预留	预留	预留
0x7000	Error _ status	见 E6.6	2
0x7001	Error _ information	用户定义	可变的
0x7002 至 0x7FFF	预留	—	—
0x8000 至 0xFFFF	用户定义		

E6.1.2 参数语义

AC _ delay _ start：当 AC 发生变化后的第一个加扰周期，这个参数用来代替 delay _ start 参数。

AC _ delay _ stop：当 AC 发生变化前的最后一个加扰周期，这个参数用来代替 delay _ stop 参数。

access _ criteria：这个参数包含 ECMG 产生 ECM 时所需要的长度、格式的 CA 系统信息，例如指向 ECMG 数据库的访问准则的指针、封装的 TLC 格式中的相关访问准则的列表等。这个参数包括在 CW _ provision 消息中指明的与 CP 相关的信息。访问准则参数是否需要及其内容是 CA 系统供应商需求决定的。

access _ criteria _ transfer _ mode：这 1 字节参数是个标志。如果为 0，表示只有当参数内容变化时需要

access _ criteria 参数。如果等于 1，表明 ECMG 要求在每个 CW _ provision 消息中包含 access _ criteria 参数。

Super _ CAS _ ID：32比特的标志符，由16比特的CA _ system _ ID和16比特的CA _ subsystem _ ID组成。对给定的 SCS，它唯一确定一个(套)ECMG。它由用户定义，属于专有。

CP _ CW _ combination：10 字节参数，由控制字、控制字相关的加扰周期组成。控制字的校验位(奇偶)和控制字所相关的加扰周期的校验位(奇偶)相等。

CP _ duration：和 nominal _ CP _ duration 不同，这个参数表示特定流的特定加扰周期的实际长度。

CP _ number：加扰周期的标志之一，表示和某个消息相关的加扰周期。它和 CW _ provision、ECM _ response 消息相关。

CW _ per _ msg：ECMG 每次请求控制字时需要的控制字的个数。如果它的值为 y，lead _ CW 为“x”，每个和加扰周期“n”相关的控制字供给消息中将包括从(n＋1＋x－y)到(n＋x)时刻之间的所有控制字。控制字通过 CP _ CW _ combination 参数与加扰周期参数一起发送。在大多数 CA 系统中，CW _ per _ msg 为 1 或 2。(见 lead _ CW 部分)例如 ECMG 请求当前的、下一个 CW 以产生 ECM，它至少需要明确一个 lead _ CW。

然而由于可以把 CW 缓存，它可以把 CW _ per _ msg 设为 1。这样，它总是接收下一个加扰周期的 CW，并从内存中取出前一个加扰周期中接收的 CW 用于当前的加扰周期。

或者可以设两个 CW _ per _ msg，这样在产生 ECM 时，两组控制字都可以用。这样减少了对 ECMG 存储的需求，而且由于每个 CW 供给消息中包含了所有的控制字，对热备份也有好处。

所有的 SCS 至少需要支持 CW _ per _ msg 为 1 或 2。

delay _ start：这个有符号整数代表加扰周期开始时刻和这个周期相关的 ECM 广播开始时刻之间的时间长度。如果是正数，表示 ECM 应根据加扰周期的开始做出延迟；如果是负数，表示 ECM 在此时间之前被广播。这个参数在 ECMG 和 SCS 建立通道时发送。

delay _ stop：这个有符号整数代表加扰周期结束时刻和这个周期相关的 ECM 广播结束时刻之间的时间长度。如果是正数，表示 ECM 的结束应根据加扰周期的结束做出延迟；如果是负数，表示 ECM 在此时间结束之前被广播。这个参数在 ECMG 和 SCS 建立通道时发送。

ECM _ channel _ ID：由 SCS 分配，唯一确定 SCS 和所有已建立连接的 ECMG 的 ECM 通道。

ECM _ datagram：由SCS 传送到MUX 的实际的ECM 消息。它根据section _ TSpkt _ flag 的值不同，可以是一系列 TS 包(188 字节长度)或 MPEG-2 段。它可以是 0 长度，表示当前的加扰周期没有 ECM 广播。ECM 数据必须符合本指导性技术文件附录 H 的规定。

ECM _ rep _ period：这个整数以 ms 为单位表示数据(如 ECM)的重复周期。

ECM _ stream _ ID：这个参数在一个通道内唯一确定一个 ECM 流。它由 SCS 在流建立之前分配。

Error _ status：错误状态。

Error _ information：这是个可选参数，它包括用户定义的数据，对 error _ status 做出补充信息。它可以是 ASCII 字符、错误参数的 ID 号等。

lead _ CW：生成 ECM 事先需要得到的控制字个数。如果它的值为 x，ECMG 需要一直和“n＋x”加扰周期相关的控制字来产生“n”加扰周期的 ECM。大多数现存的 CA 系统中，lead _ CW 为 0 或 1。见 CW _ per _ msg。

例如，ECMG 需要当前的 CW、下一个 CW 来产生 ECM，lead _ CW 则为 1，即需要下一个 CW 来产生当前的 ECM。

所有的 SCS 至少支持 lead _ CW 为 0 或 1。

max _ comp _ time：这个参数由 ECMG 在和 SCS 建立通道时发送。它表示当一个通道中所有的流在使用时，ECMG 计算 ECM 所需要的最长时间。这个时间被 SCS 用来在产生 ECM _ response 消息时，决定何时可判断为超时。这个值应该比同一个 Channel _ status 消息中的 min _ CP _ duration 参数小。

max _ streams：在一个通道中，ECMG 支持的最大可以同时打开流的数量。这个参数由 ECMG 和 SCS 建立通道时发送。它等于 0 时，表示最大数目未知。

min _ CP _ duration：这个参数由 ECMG 和 SCS 建立通道时发送。它表示 CW 改变前必须保持为有效的最短时间。这个值应该比同一个 Channel _ status 消息中的 max _ comp _ time 参数大。

Nominal _ CP _ duration：这个参数表示特定流的名义上的加扰周期。这意味着所有和这个流相关的加扰周期都取此值，除非需要事件对齐、处理错误。即使在这些额外的情况下，所有实际的加扰周期应大于或等于 nominal _ CP _ duration。而且 nominal _ CP _ duration（由 SCS 决定）至少大于等于：

1）ECMG 在建立通道时确定的 min _ CP _ duration；

2）ECMG 在建立通道时确定的 max _ comp _ time 再加上典型的网络反应时间。

section _ TSpkt _ flag：为 0 时，表示 ECM 采用 MPEG-2 段格式；为 1 时，表示采用 MPEG-2 TS 包格式。

transition _ delay _ start：在清除加扰之后的第一个加扰周期，用来取代 delay _ start 参数。

transition _ delay _ stop：在清除加扰之前的最后一个加扰周期，用来取代 delay _ stop 参数。

E6.1.3 通道专用消息

E6.1.3.1 通道建立消息

见表 E3。

表 E3 通道建立消息

参数	消息中的实例个数
ECM _ channel _ ID	1
Super _ CAS _ ID	1

Channel _ setup 消息（消息类型＝0x0001）由 SCS 在 TCP 连接建立后，建立通道时发送。它包括 Super _ CAS _ ID 参数，用以告知 ECMG 希望和哪个 CA 系统、CA 子系统建立通道。（实际上，ECMG 可以对应多个 Super _ CAS _ ID。）

E6.1.3.2 通道测试消息

见表 E4。

表 E4 通道测试消息

参数	消息中的实例个数
ECM _ channel _ ID	1

任何一方可以在任何时刻发送 Channel _ test 消息（消息类型＝0x0002），以检查：

1）通道是否无错误；

2）TCP 连接是否仍然有效。

如果通道正常，对方将返回 Channel _ status 消息；否则如果发生错误则返回 Channel _ error 消息。

E6.1.3.3 通道状态消息

见表 E5。

表 E5 通道状态消息

参数	消息中的实例个数
ECM _ channel _ ID	1
section _ TSpkt _ flag	1
AC _ delay _ start	0/1
AC _ delay _ stop	0/1
Delay _ start	1
Delay _ stop	1
transition _ delay _ start	0/1
transition _ delay _ stop	0/1
ECM _ rep _ period	1
Max _ streams	1
Min _ CP _ duration	1
Lead _ CW	1
CW _ per _ msg	1
Max _ comp _ time	1

Channel _ status 消息（消息类型＝0x0003）是对 Channel _ setup 或 Channel _ test 消息的回应。所有上述所列出的参数是必须有的。

当消息是作为对建立的回应时，参数的值等于 ECMG 请求时参数的值。在通道有效期间，对所有在其上运行的流来说，所有这些参数都是有效的。当消息作为对测试的回应时，参数的值等于通道中这些参数的暂时值。

E6.1.3.4 通道关闭消息

见表 E6。

表 E6 通道关闭消息

参数	消息中的实例个数
ECM _ channel _ ID	1

Channel _ close 消息（消息类型＝0x0004）由 SCS 发送，表明通道将要关闭。

E6.1.3.5 通道错误消息

见表 E7。

表 E7 通道错误消息

参数	消息中的实例个数
ECM _ channel _ ID	1
error _ status	1 到 n
error _ information	0 到 n

Channel _ error 消息（消息类型＝0x0005）由 ECMG 在通道发生错误时发送。

E6.1.4 流相关消息

E6.1.4.1 流建立消息

见表 E8。

表 E8 流建立消息

参数	消息中的实例个数
ECM _ channel _ ID	1
ECM _ stream _ ID	1
Nominal _ CP _ duration	1

Stream _ setup 消息（消息类型＝0x0101）由 SCS 在通道建立情况下，建立流时发送。

E6.1.4.2 流测试消息

见表 E9。

表 E9 流测试消息

参数	消息中的实例个数
ECM _ channel _ ID	1
ECM _ stream _ ID	1

Stream _ test 消息（消息类型＝0x0102）用来请求给定 ECM _ channel _ ID 和 ECM _ stream _ ID 情况下的流状态，它可以由任何实例在任何时候发送，如果没有错误对方须回应 Stream _ status 消息，如果有错误须回应 Stream _ error 消息。

E6.1.4.3 流状态消息

见表 E10。

表 E10 流状态消息

参数	消息中的实例个数
ECM _ channel _ ID	1
ECM _ stream _ ID	1
Access _ criteria _ transfer _ mode	1

Stream _ status 消息（消息类型＝0x0103） 作为对 Stream _ setup 或 Stream _ test 消息的回应。当作为对建立的回应时，取值等于 ECMG 请求时参数值。当消息作为对测试的回应时，参数的值等于

通道中这些参数的暂时值。

E6.1.4.4 流关闭请求消息

见表 E11。

表 E11 流关闭请求消息

参数	消息中的实例个数
ECM _ channel _ ID	1
ECM _ stream _ ID	1

ECM _ stream _ ID 由 SCS 在 Stream _ close _ request message 消息（消息类型＝0x0104）中指示通道中哪一个流将要被关闭。

E6.1.4.5 流关闭响应消息

见表 E12。

表 E12 流关闭响应消息

参数	消息中的实例个数
ECM _ channel _ ID	1
ECM _ stream _ ID	1

ECM _ stream _ ID 由 ECMG 在 Stream _ close _ response 消息（消息类型＝0x0105）中指示通道中的哪个流正在关闭。

E6.1.4.6 流错误消息

见表 E13。

表 E13 流错误消息

参数	消息中的实例个数
ECM _ channel _ ID	1
ECM _ stream _ ID	1
Error _ status	1 到 n
Error _ information	0 到 n

Stream _ error 消息（消息类型＝0x0106）由 ECMG 在流发生错误时发送。

E6.1.4.7 CW 供给消息

见表 E14。

表 E14 CW 供给消息

参数	消息中的实例个数
ECM _ channel _ ID	1
ECM _ stream _ ID	1
CP _ number	1

表 E14 CW 供给消息（完）

参数	消息中的实例个数
CP _ CW _ combination	CW _ per _ msg
CP _ duration	0 到 1
Access _ criteria	0 到 1

CW _ provision 消息（消息类型＝0x0201）由 SCS 发送给 ECMG，作为计算 ECM 的请求。CP _ number 参数是请求的 ECM 的加扰周期。CW 控制字包含在这个消息中，在 CP _ CW _ combination 参数中根据在通道建立时确定的 lead _ CW 和 CW _ per _ msg 的值确定的加扰周期数目。

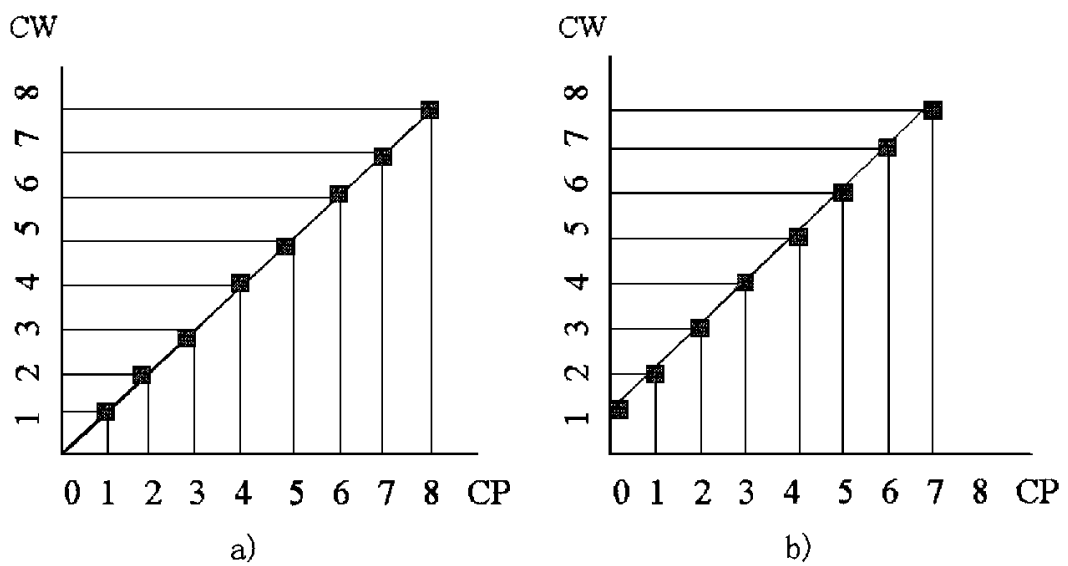
例如当 lead _ CW＝1、CW _ per _ msg＝2 时，加扰周期 N 的 CW _ provision 消息将包括N、N＋1 加扰周期的控制字。除非发生超时或者发生错误（由 SCS 厂商设备来判断），SCS 不允许在接收到前一个加扰周期的 ECM _ response 消息之前发送 CW _ provision 消息。

利用 CW _ provision 消息在 CP _ CW _ combination 中传送给 ECMG 的控制字取决于 lead _ CW 和 CW _ per _ msg 的值。表 E15 给出了这些参数满足不同的 ECMG 需求的取值情况。

表 E15 CW 供给消息参数取值示例

例子	需求	lead _ CW	CW _ per _ msg
1	1 个 CW/ECM/CP	0	1
2	每个 ECM 需要当前、下一个 CW，ECMG 从前一个 CW 供给消息得到 CW 并缓存	1	1
3	每个 ECM 需要当前、下一个 CW，ECMG 从 SCS 的每个 CW 供给消息中接收这两个 CW	1	2
4	3 个 CW/ECM/CP	1	3

图 E2 根据上面不同方式，描绘了特定 CP 必须传送的 CW 的情况。X 轴为对给定的 CP，对应的 CW 描绘在 Y 轴。在 CW _ provision 消息中，方框内的 CW 就是必须传送的 CP _ CW _ combination。



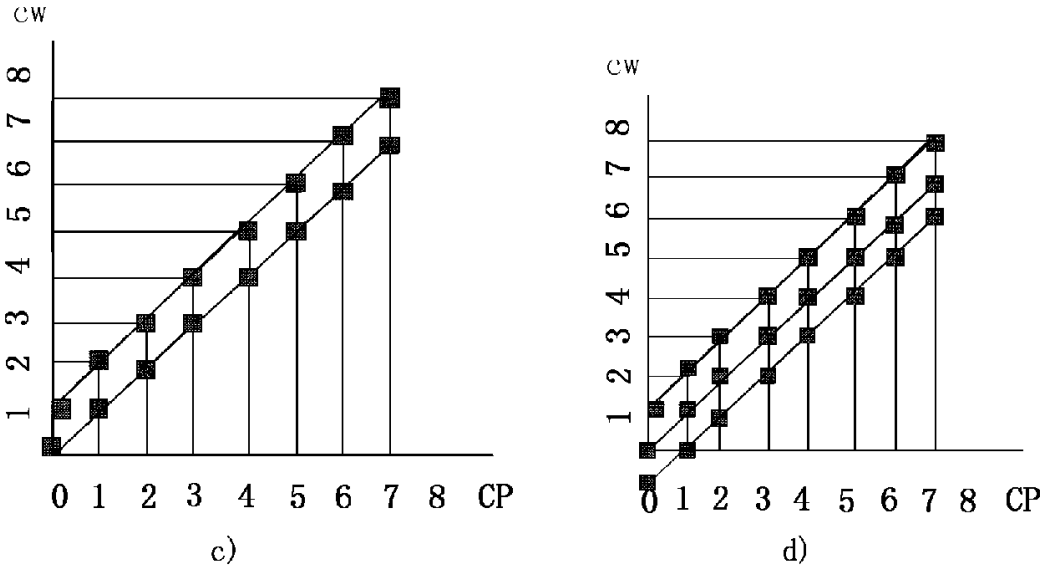


图 E2—Lead _ CW **和** CW _ per _ msg **的关系**

E6.1.4.8 ECM **响应消息**

见表 E16。

表 E16 ECM **响应消息**

参数	**消息中的实例个数**
ECM _ channel _ ID	1
ECM _ stream _ ID	1
CP _ number	1
ECM _ datagram	1

ECM _ response **消息** （**消息类型**＝0x0202）**作为对** CW _ provision **消息的回应，它承载有由** ECMG **根据** CW _ provision **消息中的信息（也可能从其他** CA **系统获得信息）计算出来的** ECM **数据。** CP _ number **值和前一个** CW _ provision **消息的** ECM _ response **消息中的** CP _ number **的值相同。**

ECM _ response **消息是否发生超时，由** SCS **根据通道建立时** max _ comp _ time、**典型的网络延时来计算得到。**

E6.1.5 **错误状态**

见表 E17。

注：本附录中未规定 TCP **连接级别的错误信息，仅仅包括了通道、流、应用级别的错误信息。这些错误一般在** TCP **连接有效的期间发生。**

在这些接口中存在两种不同的错误信息：关于通道错误的 Channel _ error **消息、关于流错误的** Stream _ error **消息。这些消息由** ECMG **发送给** SCS。**当** ECMG **向** SCS **报告错误时，由** SCS **决定采用合适的措施。**“unrecoverable error”（**“不可恢复错误”**）**表示通道或流必须关闭（取决于使用的消息）。**

下表中的错误在一般正常操作时，通常不会发生，主要是为了系统集成、调试提供方便。

表 E17 ECMG 与 SCS 接口错误码

错误状态值	错误类型
0x0000	预留
0x0001	非法消息
0x0002	不支持的协议版本
0x0003	未知的 message _ type 值
0x0004	消息过长
0x0005	未知的 Super _ CAS _ ID 值
0x0006	未知的 ECM _ channel _ ID 值
0x0007	未知的 ECM _ stream _ ID 值
0x0008	该 ECMG 通道过多
0x0009	该通道 ECM 流过多
0x000A	该 ECMG 中 ECM 流过多
0x000B	没有足够的控制字来计算 ECM
0x000C	ECMG 存储容量不足
0x000D	ECMG 计算资源不足
0x000E	未知的 parameter _ type 值
0x000D	参数长度不对
0x000F	缺少强制的参数
0x0010	参数数值无效
0x0011 至 0x6FFF	预留
0x7000	未知错误
0x7001	不可恢复的错误
0x7002 至 0x7FFF	预留
0x8000 至 0xFFFF	ECMG 规定/CA 系统规定/用户定义

E6.2 EMMG 与 MUX、PDG 与 MUX 接口

E6.2.1 参数类型值

见表 E18。

表 E18 参数类型值

参数类型值	参数类型	类型/单位	长度（字节）
0x0000	预留	—	—
0x0001	Client _ ID	Uimsbf	4
0x0002	Section _ TSpkt _ flag	Boolean	1
0x0003	Data _ channel _ ID	Uimsbf	2

表 E18 参数类型值（完）

参数类型值	参数类型	类型/单位	长度（字节）
0x0004	Data _ stream _ ID	Uimsbf	2
0x0005	Datagram	用户定义	可变
0x0006	Bandwidth	Uimsbf/kbps	2
0x0007	Data _ type	Uimsbf	1
0x0008 至 0x6FFF	预留	—	—
0x7000	Error _ status	见 E7.5	2
0x7001	Error _ information	用户定义	可变
0x7002 至 0x7FFF	预留	—	—
0x8000 至 0xFFFF	用户定义	—	—

E6.2.2 参数语义

Bandwidth：这个参数在 Stream _ BW _ request、Stream _ BW _ allocation 消息中使用，表示请求或分配的比特率。当使用带宽分配方法分配时，由 EMMG/PDG 负责在 MUX 给定的限制范围内维持这个比特率。

EMMG/PDG 将从 0kbps 到这个比特率之间工作。如果不使用带宽分配方法分配，在本附录中未规定比特率由哪部分控制。

Client _ ID：Client _ ID 是个 32 比特标志符，在多个 EMMG/PDG 连接到给定的 MUX 时，唯一确定一个 EMMG/PDG。为了使得其唯一性更加方便实现，一般使用以下原则：

在传送 EMM 或其他 CA 相关数据时，Client _ ID 的前两个字节和 CA _ system _ ID 前两个字节相同；

其他情况下，根据其应用情况分配一个值。

Data _ stream _ ID：这个标志在一个通道内唯一确定一个 EMM/专用数据流。

Data _ channel _ ID：这个标志在一个 Client _ ID 内唯一确定一个 EMM/专用数据通道。

Datagram：这里指 EMM/专用数据，根据 Section _ TSpkt _ flag 标志可以在 MPEG-2 段或 TS 包中传输。

Data _ type：在 EMM/专用数据流中传输的数据类型，对 EMM 来说为 0x00，对其他专用数据由用户定义。

Error _ status：错误状态。

Error _ information：这个可选参数包含了用于对 Error _ status 提供补充信息的用户定义数据。例如它可以是 ASCII 字符或者错误参数的 ID 值。

Section _ TSpkt _ flag：值等于 0 表示 EMM 或专用数据采用 MPEG-2 段传送，值等于 1 表示 EMM 或专用数据采用 TS 包传送。

E6.2.3 通道相关消息

E6.2.3.1 通道建立消息

见表 E19。

表 E19 通道建立消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
section _ TSpkt _ flag	1

Channel _ setup 消息（消息类型＝0x0011）由 EMMG/PDG 在 TCP 连接建立时发送到 MUX。它应该告知 MUX 正在打开通道的 EMMG/PDG 的 client _ ID。

E6.2.3.2 通道测试消息

见表 E20。

表 E20 通道测试消息

参数	消息中的实例个数
Client _ ID	1
data _ channel _ ID	1

Channel _ test 消息（消息类型＝0x0012）可以在任何时刻由任何一方发出，由另一方检查以下内容：

通道是否无错误；

TCP 连接是否仍然有效。

如果通道无错误对方须回应 Channel _ status 消息，否则回应 Channel _ error 消息。

E6.2.3.3 通道状态消息

见表 E21。

表 E21 通道状态消息

参数	消息中的实例个数
Client _ ID	1
data _ channel _ ID	1
section _ TSpkt _ flag	1

Channel _ status 消息（消息类型＝0x0013）作为对 Channel _ setup 或 Channel _ test 消息的回应。所有参数是必须的，参数值是当前通道中的暂时有效值。

E6.2.3.4 通道关闭消息

见表 E22。

表 E22 通道关闭消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1

Channel _ close 消息（消息类型＝0x0014）由 EMMG/PDG 发送，指示通道将要被关闭。

E6.2.3.5 通道错误消息

见表 E23。

表 E23 通道错误消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
error _ status	1 到 n
error _ information	0 到 n

Channel _ error 消息（消息类型＝0x0015）由 MUX 在通道发生错误时发送。

E6.2.4 流相关消息

E6.2.4.1 流建立消息

见表 E24。

表 E24 流建立消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
data _ stream _ ID	1
data _ type	1

Stream _ setup 消息（消息类型＝0x0111）由 EMMG/PDG 在通道建立时发送，以建立流。

E6.2.4.2 流测试消息

见表 E25。

表 E25 流测试消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
data _ stream _ ID	1

Stream _ test 消息（消息类型＝0x0112）用来在给定 client _ ID、data _ channel _ ID、data _ stream _ ID 的情况下请求 Stream _ status 消息。Stream _ test 可以由任何一方在任何时刻发送。对方在流无错误时回应 Stream _ status 消息，在有错误时回应 Stream _ error 消息。

E6.2.4.3 流状态消息

见表 E26。

表 E26 流状态消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
data _ stream _ ID	1
data _ type	1

Stream _ status 消息（消息类型＝0x0113）作为对 Stream _ setup 或 Stream _ test 消息的回应。参数的值是当前流中有效值。

E6.2.4.4 流关闭请求消息

见表 E27。

表 E27 流关闭请求消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
data _ stream _ ID	1

Stream _ close _ request 消息（消息类型＝0x0114）由 EMMG/PDG 发送，指示即将被关闭的流。

E6.2.4.5 流关闭响应消息

见表 E28。

表 E28 流关闭响应消息

参数	消息中的实例个数
client _ ID	1
Data _ channel _ ID	1
Data _ stream _ ID	1

Stream _ close _ response 消息（消息类型＝ 0x0115）由 EMMG/PDG 发送，指示正在关闭的流。

E6.2.4.6 流错误消息

见表 E29。

表 E29 流错误消息

参数	消息中的实例个数
client _ ID	1
Data _ channel _ ID	1
Data _ stream _ ID	1
Error _ status	1 到 n
Error _ information	0 到 n

Stream _ error 消息（消息类型＝0x0116）由 MUX 在发生流错误时发送。

E6.2.4.7 流带宽请求消息

见表 E30。

表 E30 流带宽请求消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
data _ stream _ ID	1
Bandwidth	0 到 1

Stream _ BW _ request 消息（消息类型＝0x0117）总是由 EMMG/PDG 发送，用于两种情况。如果存在带宽参数，即申请所需要的带宽；如果不存在带宽参数，即请求当前分配的带宽信息。MUX 总是以 Stream _ BW _ allocation 消息回应。

E6.2.4.8 流带宽分配消息

见表 E31。

表 E31 流带宽分配消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
data _ stream _ ID	1
Bandwidth	0 到 1

Stream _ BW _ allocation 消息（消息类型＝0x0118）用来通知 EMMG/PDG 所分配的带宽。它可以作为对 Stream _ BW _ request 消息的回应，或者作为 MUX 在带宽发生变化时的通知，它总是由 MUX 发送。如果没有带宽参数，表示所分配的带宽未知。

注：带宽分配消息可以指示和所请求的带宽不同的带宽参数（例如更少）。

E6.2.4.9 数据供给消息

见表 E32。

表 E32 数据供给消息

参数	消息中的实例个数
client _ ID	1
data _ channel _ ID	1
data _ stream _ ID	1
Datagram	1 到 n

数据供给消息由 EMMG/PDG 在给定的 data _ stream _ ID 的流上发送数据（在 EMMG 情况下为 EMM 数据）。

E6.2.5 错误状态

见表 E33。

注：本附录未规定 TCP 连接级别的错误，只包括通道、流、应用级别的错误。这些错误一般在 TCP 连接有效期间发生。

在这些接口中一般有两种错误信息，Channel _error 消息用于通道错误，Stream _error 消息用于流错误。这些消息由 MUX 发送给 EMMG/PDG。当 MUX 向 EMMG/PDG 报告错误时，由 EMMG/PDG 决定采用合适的措施。

“不可恢复错误”表示通道或流（取决于消息）将被关闭。下表中的大多数错误在正常操作时不会发生，它们主要是为了便于系统集成和调试。

表 E33　EMMG 与 SCS 接口错误码

错误状态值	错误类型
0x0000	预留
0x0001	非法消息
0x0002	不支持的协议版本
0x0003	未知的 message _type 值
0x0004	消息过长
0x0005	未知的 data _stream _ID 值
0x0006	未知的 data _channel _ID 值
0x0007	此 MUX 通道过多
0x0008	此通道数据流过多
0x0009	此 MUX 数据流过多
0x000A	未知的 parameter _type
0x000B	参数长度不对
0x000C	缺少强制的参数
0x000D	参数数值无效
0x000F to 0x7FFE	预留
0x7000	未知错误
0x7001	不可恢复的错误
0x8000 to 0xFFFF	MUX 规定 / CA 系统规定/ 用户规定

E7　时序和播出

E7.1　时序

在所有系统中，当采用控制字（CW）对数据进行加扰时，存在一个加扰周期（CP）。为了接收机能够及时重新生成 CW，ECM 必须和 CP 以正确同步时序关系播出。

为了适应不同的工作方式，SCS 负责在 CW、ECM 播出前请求足够多的 CW、ECM 数据包。图 E3 给出了 CW 产生、ECM 产生、ECM 播出、加扰周期 CP 之间的关系。

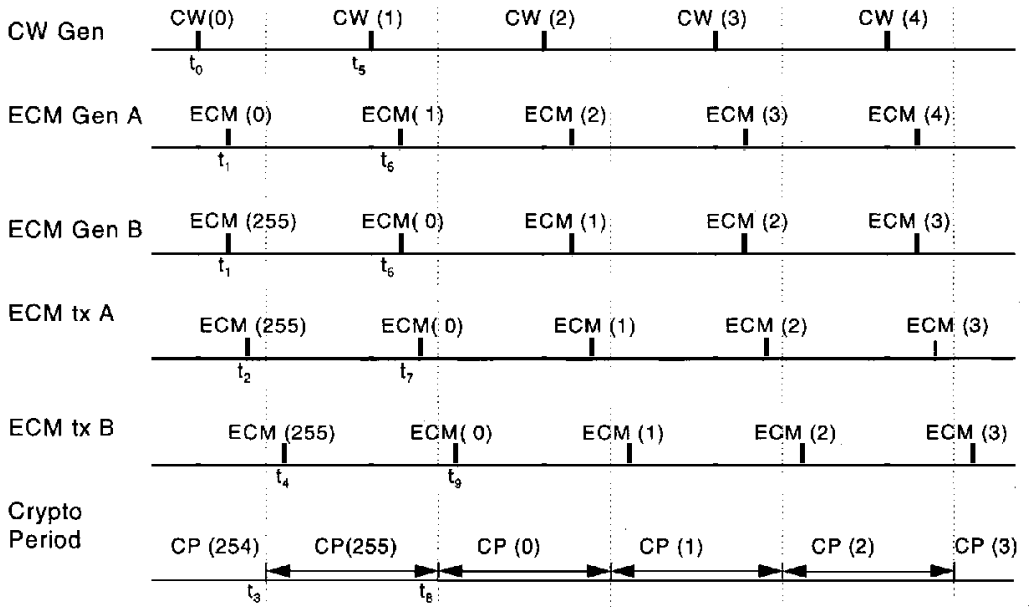


图 E3—ECM **时序图**

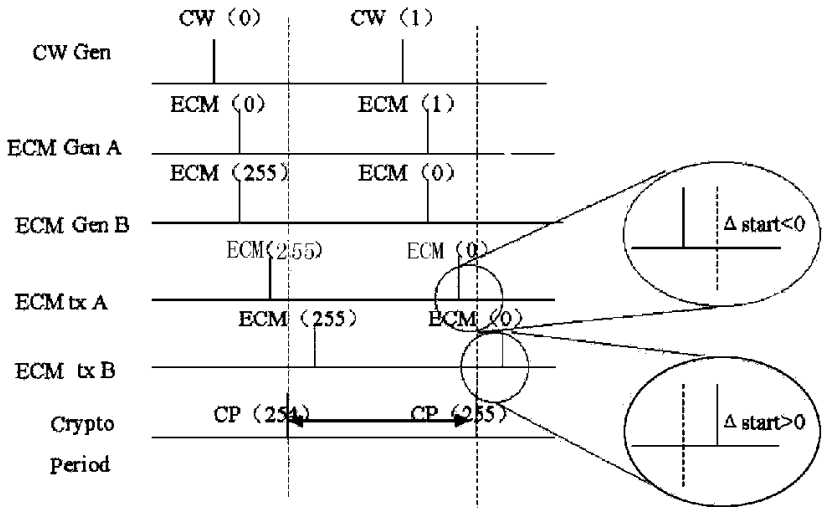
CP（0）**从** t_8 **时刻开始，加扰器采用** CW（0）**对码流加扰。每个** CA **都必须保证接收机在**CP（0）**开始前获得** CW（0）。

CA **系统** A **的** ECM **消息仅与单个** CW **相关。**ECM **生成器** A **在** t_0 **时刻收到** CW（0），**它在** t_1 **时刻生成** ECM（0），**在** CP（0）**开始前的** t_7 **时刻传送** ECM（0），**以保证接收机在** CP（0）**开始前获得**CW（0）。

CA **系统** B **的** ECM **消息与两个** CW **相关。一旦** ECM **生成器** B **在** t_0 **时刻收到** CW（0），**它就能在** t_1 **时刻生成** ECM（255），ECM（255）**包含了** CW（255）**及** CW（0）**信息。**ECM（255）**在**CP（255）**周期内的** t_4 **时刻传送。这就保证了接收机能在** CP（0）**开始前获得** CW（0）。CP（0）**开始后，在** t_9 **时刻，通过** ECM（0）**又可以获取** CW（0）**和** CW（1）。

E7.2 Delay _ start

如图 E4 **所示。**



注：图中 delay _ start 值等于 delay _ stop 的值。

图 E4 Delay _ start

Delay _ start：这个有符号整数表示加扰周期的开始时刻、与本加扰周期相关的 ECM 广播的起始时刻之间的时间差。如果是正数，表示 ECM 将根据 CP 的开始加以延迟；如果是负数，ECM 将在此时间之前广播。这个参数由 ECMG 在通道建立时向 SCS 发送。

E7.3 Delay _ stop

如图 E5 所示。

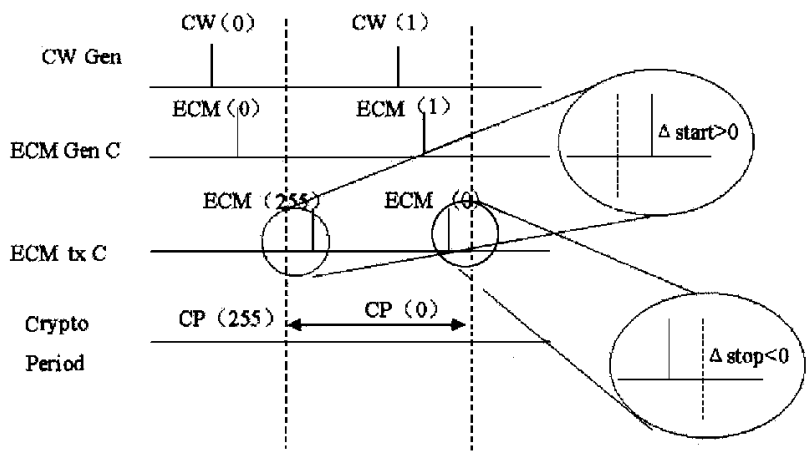


图 E5 Delay _ stop

Delay _ stop：这个有符号整数表示 CP 结尾时刻、与此 CP 相关的 ECM 广播的结束时刻之间的时间差。如果是正数，表明 ECM 广播应根据相应的 CP 的结尾加以延迟；如果是负数，表明 ECM 广播应提前结束。这个参数由 ECMG 在建立通道时向 SCS 发送。

E7.4 播出

ECM

当 ECMG 在 ECM _ response 消息中发送 ECM _ datagram 时，需注意以下两点：

a）SCS 必须根据 delay _ start 参数触发 ECM 的播发；

b）当一个新的 ECM 播发时，SCS 必须停止对同一码流的原有 ECM 的播发。

即同一节目码流中的 ECM 的播发不能交叠。SCS 还要根据 delay _ stop 参数控制 ECM 播发的终止。

如果 ECMG 出现故障（例如 SCS 在等待 ECM _ response 消息时发生超时），系统就要设法重建连接或切换到备份设备。这时 SCS 必须要延长当前的 CP，从而 ECM 的播发周期也相应延长。

EMM 和专有数据

当复用器收到 EMM 和专用数据，按照它们到达的顺序播发。

事件重新排列

如果一个新的事件在一个 CP 的中间开始，CP 就可能需要重新排列。SCS 必须保证在重新排列的过程中，没有小于 nominal _ CP _ duration 的 CP 产生。

也就是说，如果 21：00：00 有一个新事件开始，名义上的 CP 周期为 20 秒，有一个 CP 在 20：59：30 开始，那么 SCS 不能在 20：59：50 和 21：00：00 之间产生一个新的 CP。相反，SCS 应当将 CP 按照新事件的开始重新排列，它应当将前一 CP 延长到 30 秒，使它能在 21：00：00 结束。如图 E6 所示意。

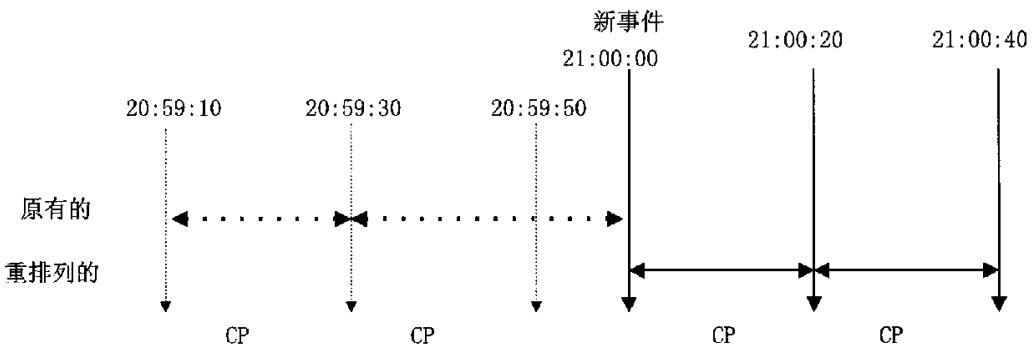


图 E6 事件重新排列

E8 系统层次

E8.1 概述

下列各节分别描述 OSI 中所定义的一个系统层次。在本附录中未描述表述层。

E8.2 物理层

物理层提供和主机之间交换数据的连接的物理工具。物理层接口应是 10/100Base-T 接口或其他完全兼容的接口。

E8.3 数据链路层

数据链路层为两个物理上直接连接的主机提供数据交换的工具（之间没有第三个主机）。物理链路层之上采用以太网协议。

E8.4 网络层

网络层在网络中具有其他主机、网关的情况下，为两个主机之间直接或间接地交换数据提供工具。网络层提供点到点的连接，应为 IP 协议。主机由其 IP 地址唯一确定。

E8.5 传输层

传输层使得两个直接或间接连接的主机可以可靠地、有次序地交换数据。而且，传输层允许一个主机上的可单独寻址的节点同一个或另外一个主机上的可单独寻址的节点之间进行通信。

TCP 使用“端口”概念和网络层提供的 IP 地址来支持可单独寻址的节点之间的通信。

E8.6 会话层

会话层向应用层提供的数据交换工具具有以下特征：

基于连接：所有的通信在两个单独定义的通信节点之间发生（非广播方式）；

按次序：接收端收到的数据次序和发送端传送的次序相同；

可靠性：可保证数据完备性，没有数据丢失；

双向：特定连接的两个通信节点都可以收、发；

无格式要求：会话层对传输的数据格式无任何强加的要求。传送给接收端的数据是无格式要求的（消息中的数据结构由应用层决定）。

应用所使用的操作系统决定了同时可以建立连接的数量。而且当发生未预见的连接关闭、连接丢失、数据读写错误时，将通知打开这个连接或进行读写操作的应用层的实例。提供这些功能的会话层应使用套接字流的接口。

E8.7 系统层次概览/通信协议栈

如图 E7 所示。

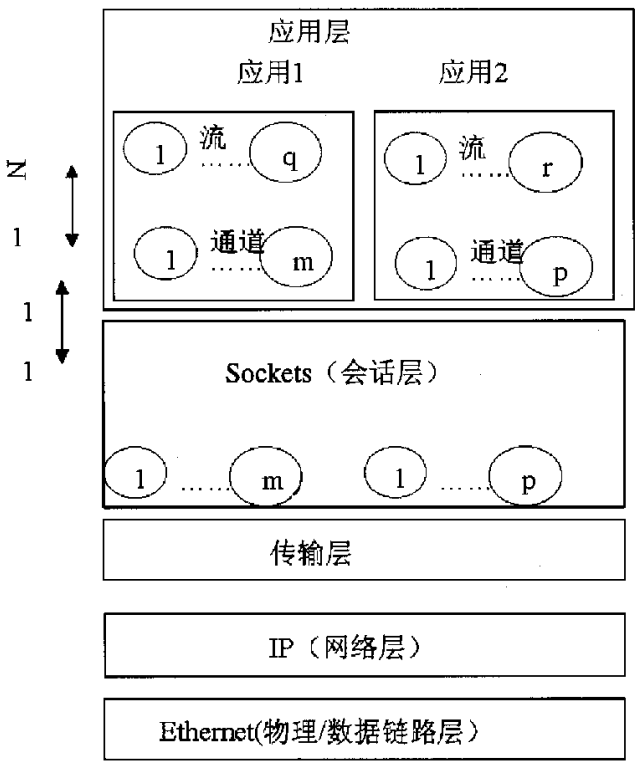


图 E7 系统层次概览图

在图的左边，描述了端口、套接字、连接、会话之间的对应关系：

“1〈—〉1”：表示一个给定类型的实例和低一层的实例一一对应关系；

“1〈—〉N”：表示可能会有多个给定类型的实例映射到低一层次的某一实例。

E8.8 建立 TCP 连接

客户机和服务器之间的连接是客户机发起的。当连接建立起来后，客户机、服务器都有一个打开的套接字，用来交换数据。服务器提供给客户机的打开连接时的 IP 地址，可以从以下两个途径得到：

静态：IP 地址信息由静态方法确定；

动态：这种方法使用 DNS（域名服务器）。客户机可以和 DNS 协商得到所需要的信息。和主机连接的 TCP 端口信息由 DNS 服务器给出，端口号一般由静态方式确定。

E9 SCS 共存

E9.1 概述

本节描述 ECMG 与 SCS 消息接口如何同另外的 SCS 通信（如在热备份配置情况下）。在同密环境下，SCS 和 MUX 通信所需要的所有信息包含在通道状态、流状态、CW 供给、ECM 响应消息中。这些信息可以传送给另外的 SCS，以维持其内部状态。

E9.2 举例

EIS 通过发送访问条件、开始时间、结束时间给 SCS＃1 触发一个 CA 事件。SCS＃1 确定本 CA 事件中包括了哪些 ECMG，并和合适的 ECMG 建立连接、通道、流。在建立过程中，每个 ECMG 通过通道和流状态消息传送所有的 ECMG 相关的消息。然后 SCS＃1 开始发送 CW 给 ECMG，在回应的数据流中接收 ECM，并同步 ECM 的播出。

在下面的例子中，SCS＃1 把从每个 ECMG 获得的消息和 CW 供给消息中的信息传送给 SCS＃2。这些消息也可以使用同样的 SCS＜＞ECMG 接口传送。主要的区别是，通常由 SCS＃1 传送给 ECMG 的 CW 供给消息在这里由 SCS＃2 从 SCS＃1 接收。这些信息允许 SCS＃2 重建 SCS＃1 上的运行环境（连接、通道、流）。见图 E8。

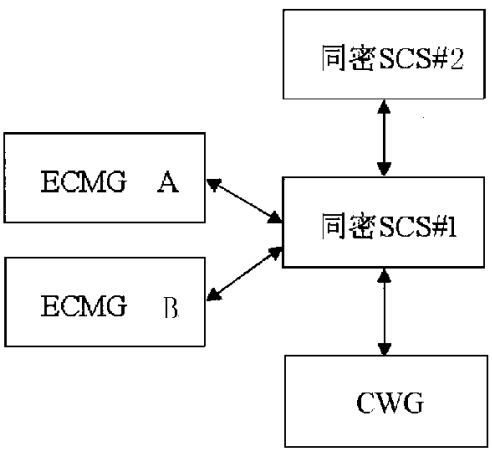


图 E8 SCS 共存示意图

E10 CW 产生和测试

E10.1 概述

控制字发生器CWG是同密系统的一部分，它负责产生用来加扰内容的加密密钥。它产生的控制字必须符合一定的随机统计特性。使用合适的（最好是物理的）的生成技术、统计测试，可以把偶然产生具有确定特性的CW的可能性降低到可以接受的程度。

E10.2 前景

产生的CW应和真正的随机序列尽可能接近。通常，产生加密学上安全的随机序列有以下要求：

a）它们必须看起来是随机的，即对观测员来说看起来拥有真随机序列的质量；

b）观测员即使拥有了所有的产生方法/硬件，也不能预测序列中下一个比特的值；

c）使用同样的输入，在同样的发生器上不能重复给定的输出序列。

使用下面的统计测试可以验证序列是否满足要求a)、b)。伪随机序列就满足这些要求。然而任何伪随机方法作为加密算法时都会受到攻击。只有满足要求a）～c）才能是真正的随机序列（加密学中所定义），在多数加密的应用中可以用做加密密钥。

这并不意味着没有伪随机技术可以用来产生同密CW，但是必须特别注意避免产生看起来是随机的但可以被非法攻破的序列。这看起来不简单，但是有一些简单的测试方法可以在开发过程中帮助它满足要求a)、b)。例如一个可接受的序列应该不可能用商业压缩程序来压缩(压缩率不大于1%～2%)。

E10.3 生成

一个产生随机序列的好方法是，使用物理现象产生平坦频谱±1 dB，(100 Hz～120 kHz）的高斯分布的白噪声。典型的，可采用热噪声、电磁辐射噪声源输入到一个高速比较器以产生数字输出。这些源容易构造，而且即使非法侵入者制造了完全相同的产生硬件也不能复制（即满足要求c)。即使采用排列组合方法，这还不能说是用线性反馈移位寄存器产生的伪随机序列。对这些方法，仍然存在一些攻击手段。

建议：同密CW发生器最好使用物理源如热噪声、二极管噪声、其他杂项或者同等的源来产生随机序列。在进行彻底的测试以保证满足要求a)、b）之后，建议可以采用伪随机技术。

E10.4 CW随机性能测试

对随机序列有多种测试方法，在实际应用中有两个基本的测试方法可用来测试伪随机和真随机序列中的明显缺陷。它们是：

1/0偏差

1/0偏差测试通常用某个合适长度的序列进行测试。比较器应调整到0的产生概率为0.5。

p（0）＝0.5＋e±0.001

这里e为偏差因子，异或后的数据按照指数规律收敛到p（0）＝0.5。

2比特情况下，一个例子是 $p(0)=0.5\pm 2e^2$

4比特情况下，一个例子是 $p(0)=0.5\pm 8e^4$

建议：

建议在产生的序列基础上进行1/0偏差测试，如果需要的话还要做出相应更改。

自相关

自相关给出了一个变量在时间上的相关程度的变化关系。为了使得序列是随机序列，其自相关特性必须最小。

E10.5 测试位置

虽然建议上面的测试在CW发生器的输出进行测试，也可以考虑在ECMG的输入端进行测试来验证接收到的CW的随机性。

附 录 F
（文件的附录）
多密技术

F1 概述

在需要对广播业务的访问条件进行控制时，需要使用条件接收系统。针对广播者可能使用不同的CA系统的情况，本指导性技术文件中提供两种解决方案：第一种为同密技术，基于各个运营者之间关于条件接收的商业协议、统一的加扰算法，这种方式适用于CA系统嵌入到接收机中的情况；第二种解决方案是在主机和模块之间定义一个标准接口，通过这个接口可以在模块中实现CA功能和更多的专有功能。这个方案允许广播者在同一个广播系统中同时使用不同厂商的模块，这样增加了广播者的选择范围，增强了反盗收的能力。本附录规定了第二种解决方案使用的公共接口。

接收机，本附录中又称为主机，具有接收未加扰的MPEG-2视频、音频、数据的功能。本附录规定了主机、解扰和CA应用之间的接口。

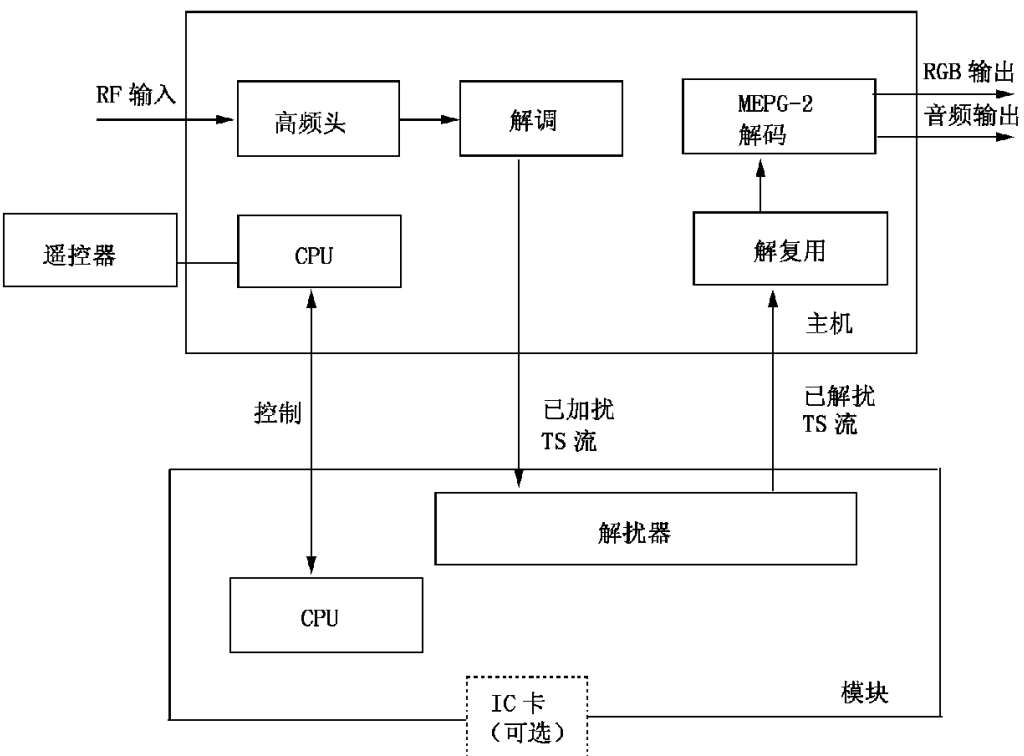


图 F1 单个模块和主机连接的示例

在同一个物理接口上定义了两个逻辑接口。第一个接口是MPEG-2 TS流接口，其链路层、物理层在本附录中定义，而更高层在MPEG-2规范中定义。第二个接口是命令接口，用以传输模块和主机之间的指令，一共定义了六层。图F1给出了模块和主机连接的例子。

本附录中只定义主机通过这个接口来实现交互所必须规定的那些方面。除了定义一整套服务以允许模块可以工作之外，对主机的设计并未做出其他限制。

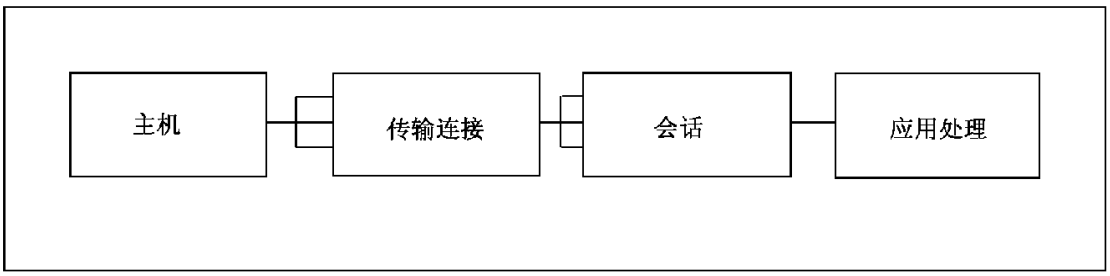
本附录中未定义条件接收系统在模块中应用的操作和功能性方面的内容。模块通过接口通信的应用并不局限于条件接收和本附录定义的内容。一个主机应可以同时支持多个模块。

F2 设计原则

F2.1 分层

本附录按层次描述，以适应将来实现的各种可能性。应用层、会话层是为公共接口上所有应用定义的。传输层、链路层可以取决于具体实现方式。在本附录中详细定义了物理接口，包括了完整的物理层规范。规范的分层结构允许灵活地使用接口，甚至超出CA应用的范围。它同时允许在一个主机上存在多个CA。

命令接口上的基本层次结构如图F2所示。主机可以和多个模块以直接或间接方式建立传输连接。当模块存在时，需要对每个连接进行维护。每个模块可以管理和主机之间的多个会话。



图F2 命令接口的层次

F2.2 物理实现

基本的规范包括在与PC卡标准兼容的物理接口上的实现方法。将来还允许其他的物理实现方法。

F2.3 客户端—服务器

接口按以下原则设计：客户端使用服务器提供的资源。应用程序运行在模块上，而资源可以由主机或主机管理的其他模块提供。由于“业务”通常指广播领域的电视、广播业务，为避免混淆，本附录中使用“资源”这个词。

F2.4 数据编码

命令接口的通信数据用“对象”来定义。对象采用ASN.1语法的通用TLV编码方法。对PC卡的实现来说，有特定的传输层编码实现，可能有其他不同的物理实现，但是语义是一致的。

F2.5 扩展性

分层结构中较高的层次设计为可扩展的。象上述的一样，TLV编码是可扩展的，以便新的对象的加入和已有对象的扩展，而不存在标志编码空间用尽或值的长度受限制这样的问题。资源管理器为CA和其他基于模块的应用提供基于主机资源的扩展机制。

F3 描述和架构

F3.1 概述

在主机中规定了一部分逻辑架构，以便于定义公共接口在何处可以和主机连接。而这些规定对主机厂商选择的自由度的影响程度已经被降低到最小。图F1是典型的主机架构、接口位置的框图。应注意的是，在一个主机上可以有多个模块存在。

公共接口包括两个部分：TS 流接口和命令接口。两个接口都采用分层结构，以使得设计、实现更加容易。分层结构中，在高层上对所有的实现是一样的，但是在低层上实现可以不同。本附录中按照 PC 卡的实有标准制定，将来的版本可以包括其他标准。

F3.2 传送流接口

传送流接口在每个方向上传输 MPEG-2 TS 包。如果模块允许接收 TS 流的业务，而且这些业务已经被主机选中，则模块返回包括这些业务已解扰的包，而其他的包并未作改变。大多数条件下，在传送流接口上，经过模块和任何物理层时都有一个固定的延迟（见本附录 F3.4 部分）。TS 流接口层如图 F3 所示。TS 层和所有的较高层在 MPEG-2 规范 GB/T 17975.1 中定义。

高层次
传输层
PC 卡链路层
PC 卡物理层

图 F3 -TS 接口层次

F3.3 命令接口

命令接口载有模块中的应用和主机之间通信的信息。为提供必要的功能，在分层结构上定义了通信协议。这些功能包括：在一个主机上支持多个模块的能力、支持主机和模块之间的复杂组合的能力、一整套对象的扩展能力，并以主机向模块提供资源的方式提供。层次结构如下图 F4 所示：

<table>
<tr><td colspan="4">应用层</td></tr>
<tr><td colspan="4">资源：</td></tr>
<tr><td>用户接口</td><td>低速通信</td><td>系统</td><td>可选扩展</td></tr>
<tr><td colspan="4">会话层</td></tr>
<tr><td colspan="4">通用传输子层</td></tr>
<tr><td colspan="4">PC 卡传输子层</td></tr>
<tr><td colspan="4">PC 卡链路层</td></tr>
<tr><td colspan="4">PC 卡物理层</td></tr>
</table>

图 F4 命令接口层次

本附录中描述的 PC 卡的实现有物理层、链路层、传输低子层。将来的物理实现可能与这些层次不同，但所有的不同应都限制于这些层。传输低子层的具体实现特性取决于编码方式、消息交换协议的具体细节、与上层之间的传输层连接标识、开始和结束方式等。会话层、资源层、应用层对所有物理实现是相同的。

应用层尽可能设计为和应用语义无关。通信是按照资源进行的，例如用户接口互操作、主机提供给模块的应用的低速通信协议等。这种策略使得模块可以更加方便地处理除了 CA 以外的其他任务。

F3.4 物理需求

F3.4.1 简介

本部分定义了物理层为实现所需功能所必须满足的条件。下面的物理层特性不是必须的，但是任何物理层的实现必须定义：主机和模块之间的机械、电连接，即套接类型和大小、针数量、电压、阻抗、功率等参数。具体的物理层特性的限制和需求如下：

a）TS 流和命令逻辑连接；

b）数据速率；

c）连接与拆除连接；

d）低级别初始化；

e）使用多个模块。

F3.4.2 数据和命令逻辑连接

物理层应支持 TS 流、命令接口的双向逻辑连接。TS 流接口可以接收 MPEG-2 TS 流，其中包括 TS 包的序列。TS 包序列是连续的或由空包分开都可。返回的 TS 包为已解扰的形式的部分 TS 包。传送流接口受到下面的限制：

a）当模块是传送流的源时，输出必须满足 GB/T 17975.9；

b）如果模块是包的源或者输入包是连续的，则每个输出包是连续的；

c）模块处理输入的 TS 包时将引入一个延迟，并受下面公式给出的最大延迟变量 $tmdv_{max}$ 的限制。

$tmdv_{max}$ =（n × TMCLKI）+（2 × TMCLKO）；

当 n=0 时，$tmdv_{max}$ <=1 ms；

这里，

tmdv=模块延迟变量；

n=对应输入 TS 包中存在的间隙数；

TMCLKI=输入数据时钟周期；

TMCLKO=输出数据时钟周期；

* 1 个间隙定义为当 MIVAL 处于非激活状态时的 1 个 MCLKI 上升沿；

* 强烈建议所有主机输出连续的 TS 包；

* 只有实现不少于 3 个 CI 接口 Socket 时，主机才能输出不连续的 TS 包；

* 包间的间隙可以变化；

d）和 CI 兼容的主机应设计为支持 Nm 个模块。Nm 是 CI 的 Socket 数量和 16 相比较两者中较大的数。它应能忍受 Nm 个模块、传送流中的抖动造成的影响。最坏情况下的抖动可以是主机自己的 Nm 个模块的输入或者 Nm－1 个模块的输入加上一个与 GB/T 17975.9 兼容的输入模块的输出；

e）所有接口在连续 TS 包的同步字节之间的平均数据传输速率应支持至少 58 Mb/s；

f）所有接口应支持最小达到 111 ns 的字节传输时钟周期。

命令接口应支持双向传输由本附录中传输层定义的命令。每个方向的数据传输速率至少支持 3.5 Mb/s。

F3.4.3 连接和拆除连接行为

无论主机是否开机，物理层应支持与模块之间的连接和拆除连接。连接、拆除连接应不会导致模块或主机的电性能损坏、不应造成模块中非易失存储器存储数据的改变。当模块未连接时，TS 流接口

应跳过该模块，该模块的命令接口应处于非激活状态。

一旦模块连接建立，主机将发起和模块的一个低级别的初始化序列。不管物理层所用的是哪个低级别的连接建立过程，这个初始化序列将建立连接并确认模块是可以兼容的。如果成功地建立连接，主机将把模块插入到主机的 TS 流路径中并建立 TS 流连接。在此过程中如果有部分 TS 流丢失也是可以接受的。同时在传输层应建立命令接口，以允许应用层初始化过程和应用层进行通信。

如果物理层用在其他应用环境中，与不兼容的模块建立连接，或者与主机相连的不是可以兼容的模块，应不会造成对模块或主机的电损坏。作为可选项，主机可以告诉用户，连接的是一个不可识别的模块。

一旦和模块的连接拆除，主机将从 TS 流数据路径中去除模块。在此过程中有部分 TS 包丢失是可以接受的。而且命令接口连接应由主机来发起拆除。

F3.4.4 多模块

应用层对主机连接的模块数量未做出任何限制。特定的物理层、特定的主机设计可以支持多个模块。虽然最小化设计的主机可以只提供一个连接，物理层规范必须允许一个主机同时连接多个模块。理想情况下，物理层规范应不对模块的数量做出限制。即使做出限制，数量应不小于 15。当主机连接有多个模块时，TS 流接口连接应采用雏菊链（Daisy-Chain）形式，如图 F5 所示。主机负责维护单独或和每个模块之间的命令接口连接，每个模块和主机之间的会话相互独立地处理。当从命令接口层连接去除一个模块，应不对其他模块产生影响。当主机同时连接多个模块时，主机应可以根据选定服务的解扰来选择相关模块。

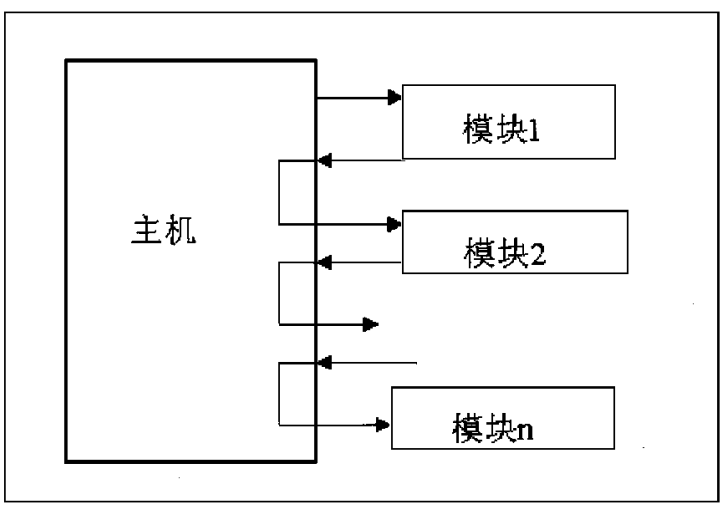


图 F5 模块间的 TS 流接口链

F3.5 操作实例

为说明上述情况，可以考虑下面的一个 PC 卡插入主机的例子。当接口的感应针感应到模块插入时，PC 卡初始化过程将开始。主机开始读取模块内存中的结构信息。这些结构信息包括模块的低级别配置信息，例如模块使用的 PC 卡读写地址，并向主机表明自己是与之可以兼容的模块。主机将关闭 TS 流接口中的跳过（bypass）链，允许 TS 包流过该模块。这将引入延迟，并导致 TS 流中短暂的间隙，但这是不可避免的。同时物理层接口初始化过程开始协商通信用的缓冲区的大小。至此物理层初始化

过程结束，对所有物理层都一样的。上层的初始化过程，在主机建立和模块的传输层连接后开始。本过程和后续的上层初始化过程在本附录的其他部分描述。

对其他物理实现来说，初始化过程逻辑上是类似的，细节上可能有所不同。

F4 传送流接口 TSI

F4.1 TSI - 物理、链路层

这些层取决于模块的具体物理实现。

F4.2 TSI - 传输层

传输层和 MPEG-2 系统的传输层的应用相同。数据以 MPEG-2 TS 包的形式在 TS 流接口上传送。整个复合的 MPEG-2 流在 TS 流接口上传送，并返回全部或部分已解扰的流。如果包未加扰，则模块按原样返回。如果 TS 包是加扰的并且包属于已选定的服务，而且模块也能够允许接收这个服务，那么模块将返回对应的已解扰的包，并把传输加扰控制标志 transport _ scrambling _ control 设为 00。

如果是在 PES 级别上加扰，模块按照同样的方式在同样的条件下工作，返回对应的解扰 PES 包，并设置 PES 加扰控制标志 PES _ scrambling _ control 设为 00。

TS 包和 PES 包在 GB/T 17975.1 中有定义。

F4.3 TSI - 更高层

除 PES 外，TS 流层上的任何层次或结构的 MPEG-2 数据和本附录无关。本附录假定模块可以从 TS 流中寻找并提取所需要的数据，例如 ECM、EMM 信息。

F5 命令接口—传输和会话层

在命令接口上通信的数据按照对象的形式定义，对象采用符合 ASN.1 语法的通用 TLV 编码方式编码。

表 F1 传输、会话、应用层协议数据单元使用的长度域

句法	比特数	类型
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
}		
}		
}		

本部分描述在命令接口传输层、会话层上传输的 ASN.1 对象。对所有的这些对象，以及 F6 中的应用层对象，表 F1 中的长度域都适用，以表明随后的域的字节数。

Size _ indicator 是长度域的第一个比特，如果等于 0，则数据域的长度为随后的 7 比特中编码。

0—127 之间的任何长度可以在 1 字节内编码。如果长度超出 127，则 Size _ indicator 等于 1。这种情况下，后续的 7 比特表示长度域的后续字节数。再后续的字节数将合并到一起，以第一个字节在最前面的形式，以整数值形式编码表示总长度。长度为 65535 以下的值可以用 3 个字节编码。在此并未使用 ASN.1 中的不确定长度格式的编码规则。

F5.1 通常的传输层

F5.1.1 概述

命令接口传输层在基于不同具体物理实现提供的链路层之上工作。对 PC 卡的物理实现来说，链路层在目前已标准化的附录 A 中详细描述。传输协议总是假定链路层是可靠的，即数据按照正确顺序传送，并不存在数据删除或重复现象。

传输协议采用命令—响应的形式进行。主机采用命令传输协议数据单元 C _ TPDU 的形式向模块发送命令，等待模块采用响应传输协议数据单元 R _ TPDU 形式进行响应。模块不能初始化通信过程，它必须等待主机查询或者先发送数据。传输协议一共由 11 个传输层对象支持。其中一部分只在主机的 C _ TPDU 中出现，另一些只在模块的 R _ TPDU 中出现，有些则在两部分中都出现。Create _ T _ C、C _ T _ C _ Reply，建立新的传输连接；Delete _ T _ C、D _ T _ C _ Reply，清除连接；Request _ T _ C、New _ T _ C 允许模块请求主机产生一个新的传输连接；T _ C _ Error 用于发送错误信息；T _ SB 载有从模块到主机的状态信息；T _ RCV 请求等待接收从模块来的数据；T _ Data _ More、T _ Data _ Last 从上层在主机和模块间传输数据，带有空数据域的 T _ Data _ Last 用于主机在自己没有数据发送时向模块查询数据。

主机的 C _ TPDU 只包含一个传输协议对象。模块的 R _ TPDU 可以有一个或两个传输协议对象。R _ TPDU 的单个对象、或者两个对象中的后一个总是 T _ SB 对象。

F5.1.2 传输协议对象

所有的传输层对象包括一个传输连接标志。此标志为一个字节，允许最多同一个主机同时有 255 个传输层连接处于激活状态。传输连接标志值 0 为预留值。这个标志值总是由主机确定。由于对所有物理实现，协议是相同的，这里将详细描述。但对象只做简单描述，详细的对象的描述和采用的具体物理层实现有关。

对每个模块支持的套接字（Socket），主机应允许 16 个传输连接。最好所有的 255 个传输连接在模块套接字中分布。

a）Create _ T _ C 建立传输连接。它只由主机发起，并包含所建立连接的传输连接标志；

b）C _ T _ C _ Reply 是目标模块对 Create _ T _ C 的响应，并含有所建立连接的传输连接标志；

c）Delete _ T _ C 删除已存在的传输连接。它含有要删除的传输连接的传输连接标志。它可以由主机或模块发起。如果由模块发起，则以对主机的查询或数据的响应来实现；

d）D _ T _ C _ Reply 是对 Delete _ T _ C 的响应。在某些情况下，这个响应可能不能到达接收目标方，因此 Delete _ T _ C 对象有个相关的超时值。如果在响应到达目标前时间超时，所有在接收到响应后应采取的操作都可以进行；

e）Request _ T _ C 请求主机产生一个新的传输连接。它由存在传输连接的模块发出。它以对主机的查询或数据做出响应的方式实现；

f）New _ T _ C 是对 Request _ T _ C 的响应。它在和 Request _ T _ C 对象同样的传输连接上传输，载有新的连接的传输连接标志。New _ T _ C 之后紧跟着新连接的 Create _ T _ C 对象，用来建立合适的传输连接；

g）T_C_Error 用来告知错误信息，并含有 1 字节的错误代码来描述错误信息。在本版本中，仅仅用来作为对 Request_T_C 的响应，表示再没有可用的传输连接；

h）T_SB 作为对主机所有对象的响应，可以附加在其他协议对象之后，也可以单独发送。它长度为 1 字节，表示模块是否有数据要发送；

i）T_RCV 由主机发送，请求模块想要发送的数据（由模块的前一个 T_SB 对象指示）；

j）T_Data_More、T_Data_Last 含有主机、模块之间的数据，可以包含在 C_TPDU 或 R_TPDU 中。对模块来说，它们仅仅作为对主机请求的响应来传送。T_Data_More 在由于外部对数据大小限制的情况下，需要对上层的协议数据单元 PDU 分裂成多个块时使用。它表示在此之后至少有 1 个上层 PDU 的块需要传送。T_Data_Last 表示这是上层 PDU 的最后一个或者唯一的块。

F5.1.3 传输协议

图 F6、F7 描述了主机、模块之间连接建立和拆除的状态变化过程图。每个状态变化用导致这个变化的事件来标志。如果这个变化同时导致需要发送新的对象，则用方框内的文字表示。表 F2、F3 描述了期望的对象。

当主机想建立和模块间的传输连接，它发送 Create_T_C 对象，并把状态转移到“正在建立”。模块采用 C_T_C_Reply 对象来响应。如果模块响应超时，主机返回到空闲状态（通过“超时”条件）。主机对此传输连接将不再传送数据或查询，后来的 C_T_C_Reply 也将被忽略。如果后来主机重新使用相同的传输连接标志，模块将再次收到 Create_T_C，表示旧的连接已经无效，需要建立新的连接。

当模块用 C_T_C_Reply 响应主机的请求，主机的状态变化到“有效”状态。如果主机有数据需要发送，主机现在可以发送。否则它则需要按照常规方式查询连接才能发送。

如果主机想结束一个传输连接，它将发送 Delete_T_C 对象，状态变化到“正在删除”。主机收到 D_T_C_Reply 对象后，或者发生超时情况时，将返回到“空闲”状态。如果主机收到模块发送的 Delete_T_C 对象，它发出 D_T_C_Reply，并直接变化到“空闲”状态。除了处于“有效”状态外，任何状态下预期之外的接收到的对象将被忽略。

在“有效”状态下，主机周期性地查询，或者在有上层的 PDU 需要发送时发送数据。作为响应，如果模块想要请求或删除传输连接，主机将收到从模块发送的 T_SB 对象、以及在此之前的 Request_T_C 或 Delete_T_C 对象。在“有效”状态下，主机在任何时刻可以发送数据。如果模块想要发送数据，它必须等待主机的消息（通常是数据或查询），然后用 T_SB 表示自己有数据要发送。主机然后在某个时刻（不必是马上）发送一个 T_RCV 请求给该模块，模块在 T_Data 中发送数据作为响应。如果使用 T_Data_More，每个后续的块前需要等待主机的下一个 T_RCV 消息才能发送数据。

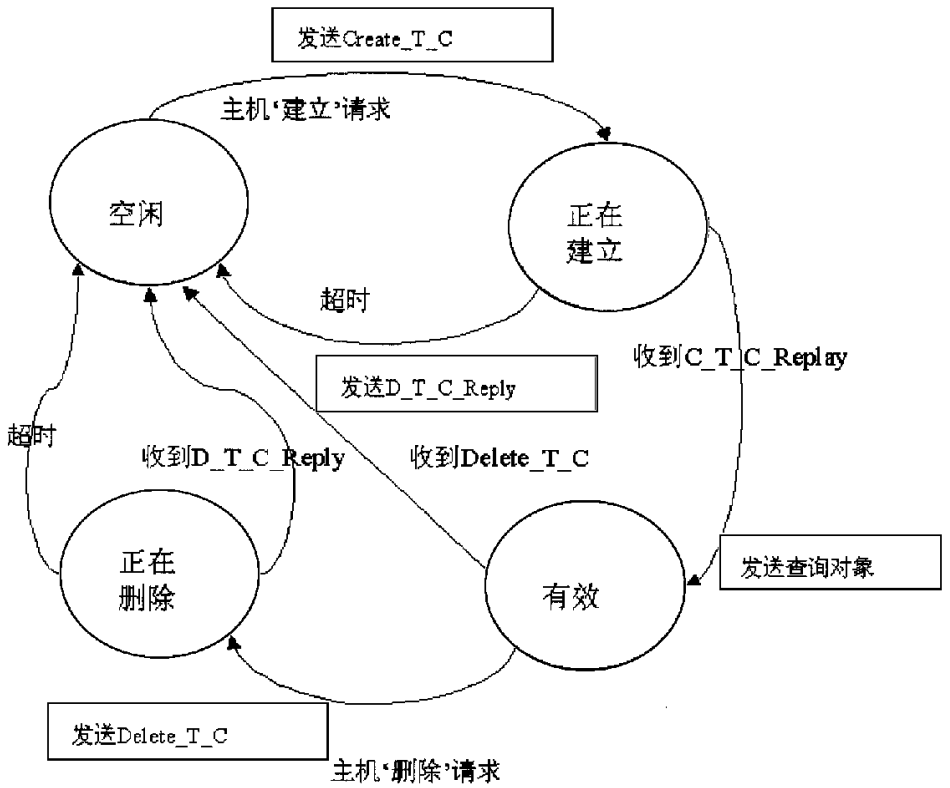


图 F6　传输协议的主机端状态转移图

表 F2　主机传输连接状态下期望收到的对象

状态	主机预期的对象
空闲	无
正在建立	C_T_C_Reply（＋T_SB）
有效	T_Data_More，T_Data_Last，Request_T_C，Delete_T_C，T_SB
正在删除	D_T_C_Reply（＋T_SB）

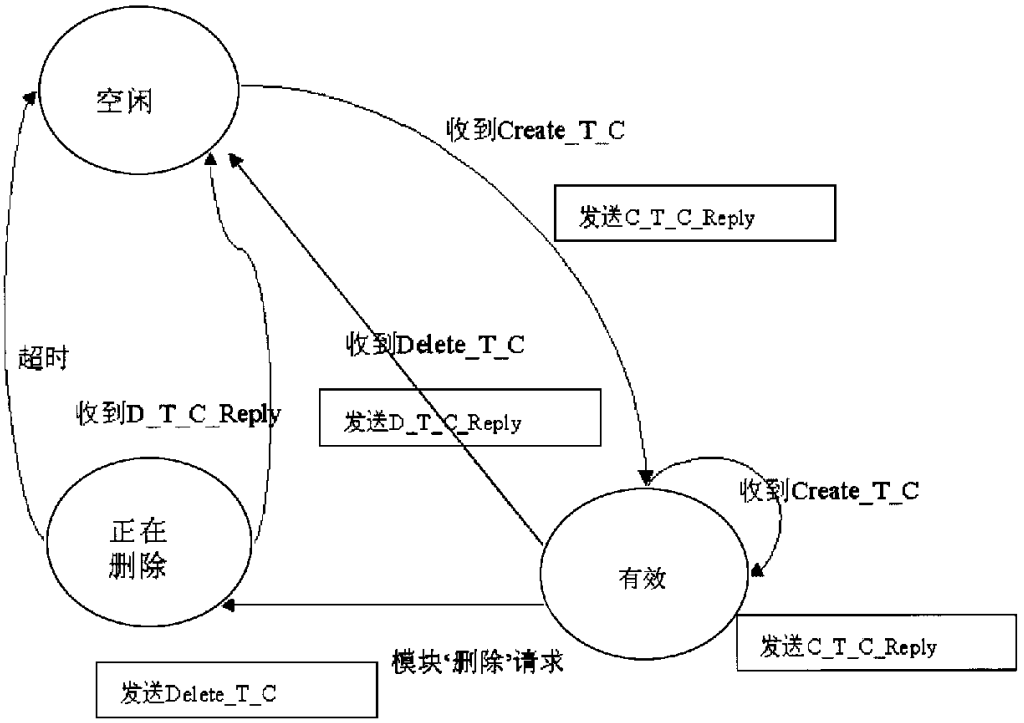


图 F7　传输协议的模块端状态转移图

表 F3　模块传输连接状态下预期收到的对象

状态	模块预期收到的对象
空闲	Create _ T _ C
有效	Create _ T _ C,T _ Data _ More,T _ Data _ Last,New _ T _ C, Delete _ T _ C,T _ RCV,T _ C _ Error
正在删除	D _ T _ C _ Reply

如果模块想建立另一个传输连接，它应发送 Request _ T _ C 作为对主机数据或查询的响应。如果满足条件，主机将以包含新连接的传输连接标志的 New _ T _ C 作为响应，并在其后紧跟 Create _ T _ C 对象以建立连接。如果主机由于所有的传输连接标志正在使用而不能满足需求，它将以包括合适错误代码的 T _ C _ Error 作为响应。

图 F8 给出了产生和使用传输连接的对象转移序列的情况。

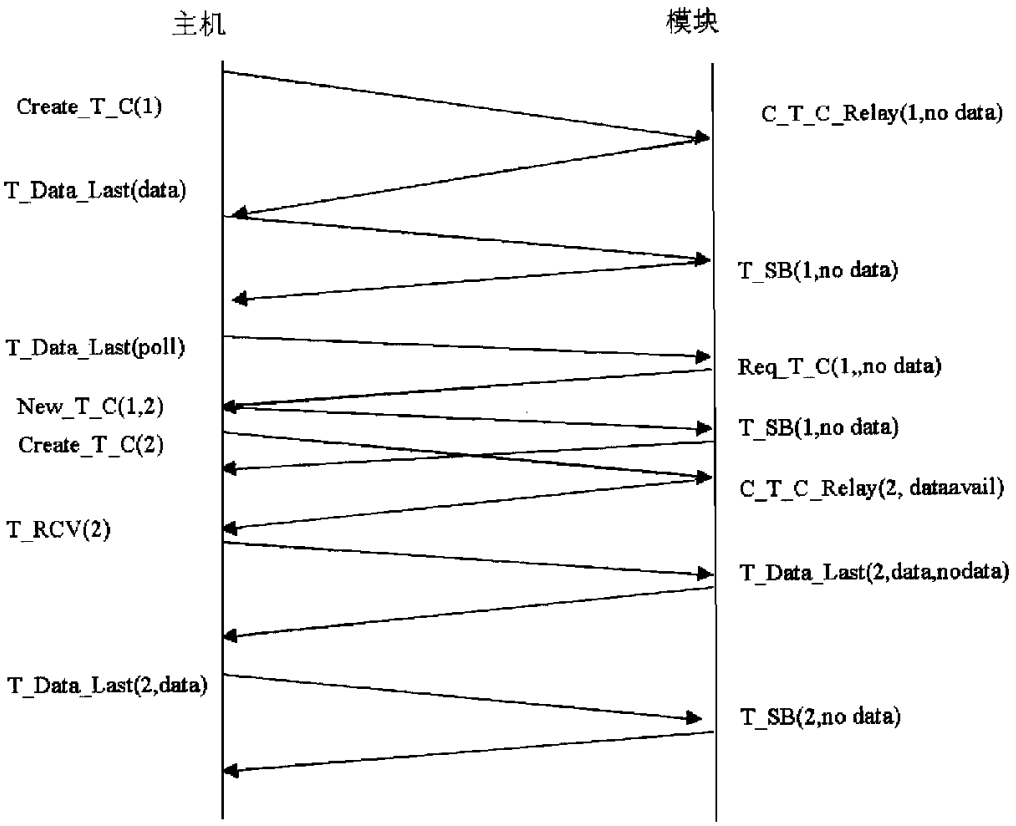


图 F8　对象转移序列的例子

在这个例子中，假定模块刚刚被插入到主机中并建立了物理连接。主机发送 Create _ T _ C 以建立传输连接 1（如果已经插入其他模块，这将是更大的数）。模块马上以 C _ T _ C _ Reply 作为对传输连接 1 的响应，表示没有数据需要发送。主机发送一些数据以及 T _ Data _ Last，模块以 T _ SB 响应表示没有数据需要发送。稍后主机以空的 T _ Data _ Last 对模块查询，模块以 Request _ T _ C 响应，表示模块想建立一个新的传输连接，同时表示（在附加的 T _ SB 中）模块在连接 1 上没有数据要发送。

主机以 New _ T _ C 响应，表示即将建立传输连接 2。在此后紧跟传输连接 2 的 Create _ T _ C。模块 T _ SB 对第一个、C _ T _ C _ Reply 对第二个做出响应同时表示在此连接上没有数据要发送。主机以 T _ RCV 作为响应来接收数据，模块以包含数据的 T _ Data _ Last 作为响应。主机以自己的数据作为响应，模块做出响应并指示没有进一步的数据需要发送。两个传输连接都会保持激活，直到模块或主机删除它为止，并且主机将周期性地用空的 T _ Data _ Last 进行查询。

F5.2　会话层

F5.2.1　概述

会话层为应用提供利用资源的机制。资源是一种在应用层封装功能的机制，将在本附录 F6.2 详细描述。资源同时支持的会话数量可以不同。

有些资源支持一个会话。如果第二个应用企图请求一个会话正在使用的资源，将得到“资源忙”的响应。其他资源可以同时支持多个会话，资源请求必须受到资源自身所定义的条件的限制。后者的一个典型例子是显示资源，支持在不同显示窗口中同时显示。

F5.2.2　会话协议对象

会话对象作为协议在本部分进行描述，对象的详细编码将在后续部分描述。

open _ session _ request 打开会话请求：由应用在传输连接上发送给主机，请求使用资源。主机可以直接提供这个资源，也可以通过另一个模块提供这个资源（通过第二个传输连接）。后一种情况下，主机使用 create _ session 对象通过合适的传输连接对会话进行扩展；

open _ session _ response 打开会话响应：由主机发给请求资源的应用作为响应，以分配一个会话号，或者告诉模块这个请求不能完成；

create _ session 建立会话：由主机发给模块中的资源提供者，以扩展另一个传输连接上的应用的会话请求；

create _ session _ response 建立会话响应：由模块中的资源提供者发出给主机，这样主机可以告诉发起的模块此会话是否可以打开；

close _ session _ request 关闭会话请求：由模块或主机发起，以关闭一个会话；

close _ session _ response 关闭会话响应：由模块或主机发出，告知会话已关闭；

session _ number 会话号：总是在包含 APDU 的 SPDU 前面。

F5.2.3 会话协议

会话层的对话由主机或模块发起。下面给出了两个例子。在第一个例子中，模块 A 想使用主机提供的资源。见图 F9。第二个例子中，模块 A 想使用模块 B 提供的资源。见图 F10。

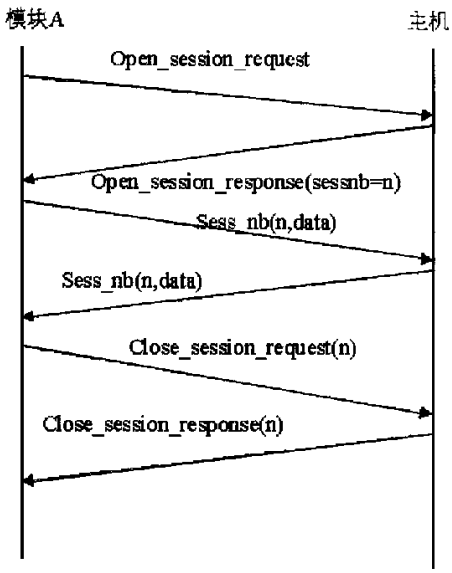


图 F9 主机提供资源的会话

模块 A 请求在传输连接上打开一个会话以使用资源。由于主机自身包括资源，它在响应消息中将直接包括会话号。这样通信可以开始了。在应用层数据之前加上会话号。最后会话被关闭，这个例子中由模块 A 发起关闭请求。会话也可以由主机关闭，例如当由于某种原因导致资源不可用时。

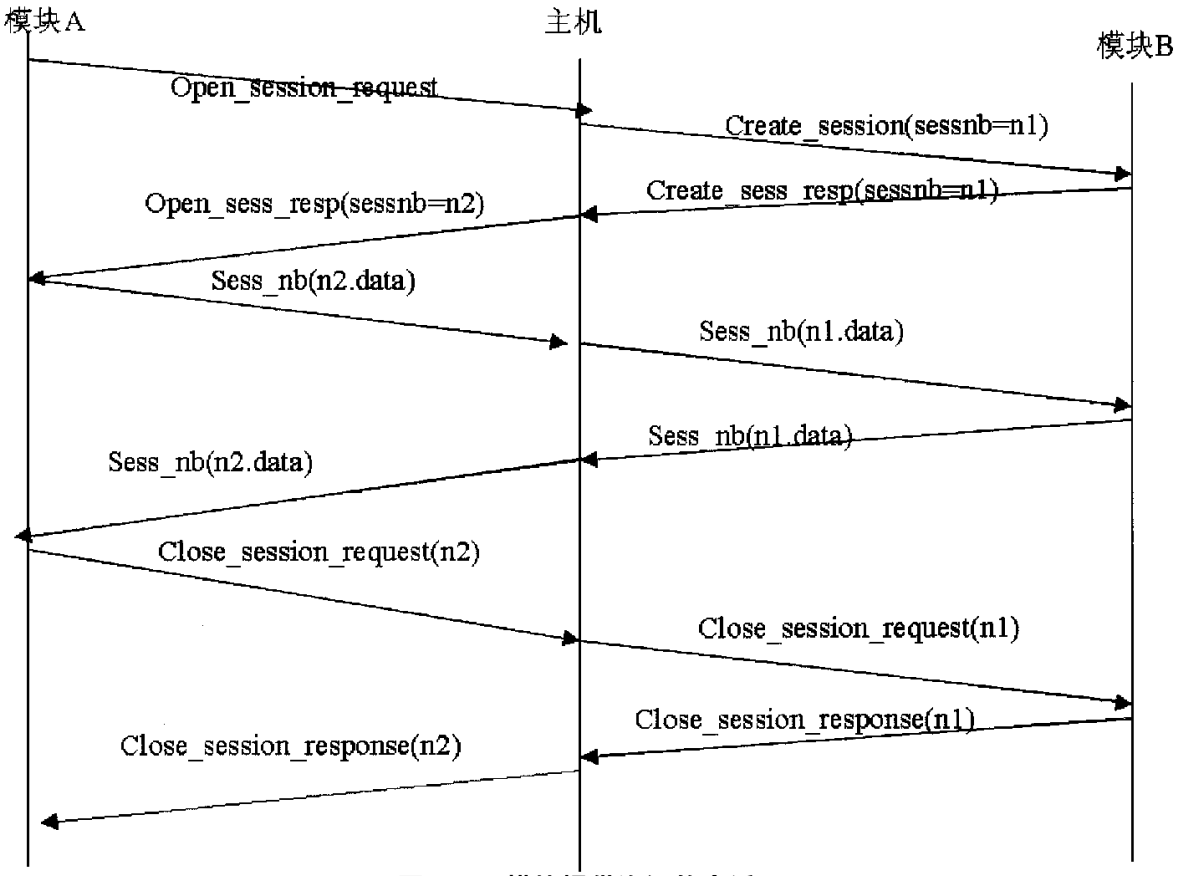


图 F10　模块提供资源的会话

模块 A 在传输连接上请求打开会话以使用资源。主机知道这个资源是模块 B 提供的，因此主机用建立会话对象来扩展资源提供者的传输连接上的会话。资源提供者对此做出响应，主机再对原先的请求者做出响应。

应用层数据的通信现在可以开始了。主机在输入、输出的传输连接间进行必要的路由。最后会话被关闭，这个例子中由模块 A 发起，但它也可以由主机或模块 B 发起。会话 n1、n2 是由主机分配的，这个例子中它们是不同的。在同一个传输连接上使用相同还是不同的会话号，在主机具体实现时是可选的。应用层在会话的任何一端都不能假定会话号是相同或者不同的。

F5.2.4　SPDU 结构

在会话级别上，会话层使用会话协议数据单元 SPDU 来在主机和模块之间交换数据。图 F11 和表 F4 给出了 SPDU 的结构和编码。

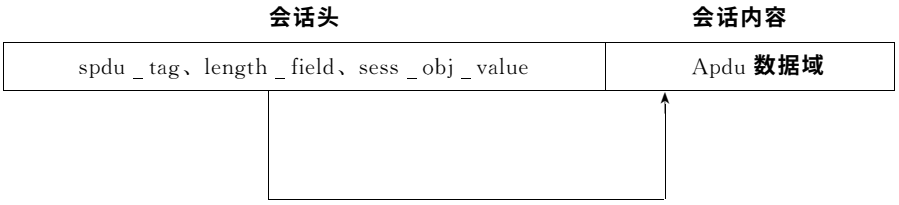


图 F11 SPDU **结构**

表 F4 SPDU **编码**

句法	比特数	类型
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 ()		
}		
}		

SPDU **包括两部分**：

— **一个强制包含会话头，由标识值** spdu _ tag、**表示会话长度的** length _ field、sess _ obj _ value **组成。会话对象在** F5.2.6 **描述。应注意的是，长度域不包括后面的** APDU **部分**；

— **一个长度可变的会话内容，包含属于同一个会话的整数个** APDU。**是否有会话内容取决于会话头**。

F5.2.5 SPDU **的传输**

一个 SPDU **在一个或几个** TPDU **的数据域内传输**。TPDU **可以见物理模块实现部分的描述**。

F5.2.6 **会话头描述**

一个 SPDU **头后面跟着数据域（会话号对象），会话号后面总是跟着包含一个或几个** APDU **的** SPDU **的内容**。

F5.2.6.1 **打开会话请求**

这个对象由模块发送给主机，请求打开模块和主机、或另一模块提供的资源之间的会话。资源标志必须和主机中的当前可用资源的类型、级别完全一致。如果提供的资源标志的版本域为 0，**主机将使用自己列表中的版本号。如果请求中的版本号和主机列表中的版本号相同，将使用当前的版本。如果请求中的版本号比主机列表中的版本号大，则主机将拒绝请求，并返回合适的错误代码。见表** F5。

表 F5 打开会话请求编码

句法	比特数	类型
open _ session _ request () {		
open _ session _ request _ tag	8	uimsbf
length _ field () =4		
resource _ identifier ()		
}		

F5.2.6.2 打开会话响应

主机向模块发送这个对象，以分配一个会话号或者告诉模块它的请求不能得到满足。见表 F6。

表 F6 打开会话响应编码

句法	比特数	类型
open _ session _ response () {		
open _ session _ response _ tag	8	uimsbf
length _ field () =7		
session _ status	8	uimsbf
resource _ identifier ()		
session _ nb	16	uimsbf
}		

session _ status 会话状态

取值见表 F7。

表 F7 打开会话状态取值

会话状态	session _ status 值（十六进制）
会话已打开	00
会话未打开，资源不存在	F0
会话未打开，资源存在但不可用	F1
会话未打开，资源存在但版本比请求的版本低	F2
会话未打开，资源忙	F3
其他	预留

resource _ identifier 资源标志

主机返回所请求的资源的实际 resource _ identifier 资源标志值、以及当前的版本号。如果响应为“资源不存在”，则 resource _ identifier 域的取值将和打开请求中所提供的值相同。

session _ nb 会话号

主机为所请求的会话分配的会话号。0 是预留值。在后续的模块和资源之间的 APDU 的交换过程（通过主机）中将使用 session _ nb，一直到会话结束为止。当会话不能被打开（session _ status 不等于

0）时，session _ nb 值没有意义。

F5.2.6.3 建立会话

这个对象由主机发给提供资源的模块，以请求打开一个会话。见表 F8。

表 F8 建立会话编码

句法	比特数	类型
create _ session () {		
create _ session _ tag	8	uimsbf
length _ field () =6		
resource _ identifier ()		
session _ nb	16	uimsbf
}		

resource _ identifier

表示所请求的资源的资源标志，以当前的版本号为结尾。

session _ nb

表示主机为新会话分配的会话号。

F5.2.6.4 建立会话响应

这个对象由提供资源的模块发送给主机，以告知会话是否可以打开。见表 F9。

表 F9 建立会话响应编码

句法	比特数	类型
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

和 open _ session _ response 使用相同的状态值，见表 F7。

resource _ identifier

模块在 create _ session _ response 中插入请求的资源的类型、级别，但版本号采用模块当前的版本号。

session _ nb

和 create _ session 中的 session _ nb 域的值相同，作为对其的响应。

F5.2.6.5 关闭会话请求

这个对象由模块或主机发起用于关闭会话。见表 F10。

表 F10 关闭会话请求编码

句法	比特数	类型
close _ session _ request () {		
close _ session _ request _ tag	8	uimsbf
length _ field () =2		
session _ nb	16	uimsbf
}		

F5.2.6.6 关闭会话响应

这个对象由模块或主机发起，以确认会话的关闭。见表 F11。关闭会话状态取值见表 F12。

表 F11 关闭会话响应编码

句法	比特数	类型
close _ session _ response () {		
close _ session _ response _ tag	8	uimsbf
length _ field () =3		
session _ status	8	uimsbf
session _ nb	16	uimsbf
}		

表 F12 关闭会话状态值

会话状态	会话状态值（十六进制）
应要求，会话被关闭	00
请求中 session _ nb 未分配	F0
其他	预留

F5.2.6.7 会话号

这个对象总是在包含 APDU 的 SPDU 内容的前面。见表 F13。

表 F13 会话号编码

句法	比特数	类型
session _ number () {		
session _ number _ tag	8	uimsbf
length _ field () =2		
session _ nb	16	uimsbf
}		

F5.2.6.8 会话标记标号

表 F14 给出了命令接口会话层使用的对象的名称。spdu _ tag 的编码符合 ASN.1 的要求。每个

spdu _ tag 编码为 1 字节。

表 F14 会话标记值

spdu _ tag	tag 值（hex）	原始（P）或构建（C）的	方向（主机<——>模块）
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	<———>

80—8F，90—9F，A0—AF，B0—BF 等其他范围内的值为预留值。

F6 命令接口应用层

F6.1 概述

应用层使用一整套基于资源的协议。资源定义了模块上运行的应用可以使用的功能单元。每个资源支持一套对象、以及用于对象交换的协议。和资源之间的通信是通过和这个资源相关的会话实现的。本部分将描述所有主机必须具有的最小的资源集合。

F6.2 资源

F6.2.1 介绍

资源可以由主机直接提供，也可以驻留在其他模块中。资源由资源标志识别，资源标志包括：资源级别、资源类型、资源版本号。资源级别定义了一套对象以及使用这些对象的协议。资源类型在一个级别中定义了明确的资源单元。同一级别内所有的资源类型使用相同的对象、协议，但提供不同的服务，或者属于同一服务的不同实例。资源版本允许主机在级别、类型相同，多于一个版本存在的情况下识别最新的版本。这允许使用最新的或增强的资源来取代主机或另一模块上的资源。版本较高的资源应和先前的版本兼容，这样采用先前版本的应用仍然可以有预期的行为。

资源由和资源之间建立会话的应用所使用。通过由资源管理器管理的初始化过程，主机将识别所有的可用资源，并把会话和合适的资源相联系，而不管这些资源是自己提供的还是其他模块提供的。一旦会话建立，应用可以根据定义的协议交换对象使用相应资源。关于使用资源的一个例子是低速通信资源级别。它将提供一个使用串行通信连接——典型为调制解调器或者有线网的回传——的通用机制。资源类型根据不同速度的调制解调器，如 ITU-T 建议的 V.21、V.22、V.32bis 规定。现在大多数的调制解调器提供一个速率范围，因此一个调制解调器可以属于不同的资源类型，可以由和特定资源类型相关的会话来选择速度和类型。

F6.2.2 资源标志

资源标志包括 4 个字节。第一字节前两比特表示表示资源是公用的还是专用的，从而决定剩余部分的结构。取值 0、1、2 表示是公用资源；取值 3 表示是专用资源。

公用资源类型取值范围为1—49150。resource _id _type 域作为 resource _class 资源级别的最高比特位。0 是预留值，最大（全1）值也是预留的。专用资源通过专门定义专用资源者来标志。每个专用资源定义者可以定义 private _resource _identity 专用资源标志的结构、内容。除了全1为预留外，可以任意选择。见表 F15。

表 F15 resource _identifier 资源标志编码

句法	比特数	类型
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
}		
}		

F6.2.3 应用和资源提供者

在同一模块中可以驻留有两种类型的应用层实例。应用是利用资源来完成相关任务的。资源提供者提供主机上现有资源的附加资源、可替代主机先前版本的新版本的资源、以及其他模块提供的资源。为避免死锁问题、降低初始化的复杂度，资源提供者应不依赖于除了资源管理器外的其他任何资源是否存在来提供相关的资源。

F6.3 应用协议数据单元

F6.3.1 介绍

在模块与主机之间、模块之间来传送应用数据时，应用层的所有协议使用通用的 APDU 结构。图 F12、表 F16 给出了 APDU 的结构和编码。

头	内容
apdu _tag、length _field	[数据域]

图 F12 APDU 结构

表 F16 APDU 编码

句法	比特数	类型
APDU () {		
apdu _ tag	24	uimsbf
length _ field ()		
for (i=0; i<length _ value; i++) {		
data _ byte	8	uimsbf
}		
}		

APDU 由两部分组成：

一个头，由表明在数据域中所传送参数的 apdu _ tag、表示后续数据的长度的 length _ field 组成。apdu _ tag 和 length _ field 必须按照 ASN.1 规则编码。

一个长度等于 length _ field 长度域的数据内容部分。APDU 内容在以下章节中介绍，在将来的版本中还可以扩展。

F6.3.2 APDU 数据域链

如果接收主机或模块的缓冲区需要，APDU 内的数据可以分裂成几个较小的块。这种机制只对一些 APDU 可用。APDU 数据域链使用两种不同的 apdu _ tag 值（M _ apdu _ tag 和 L _ apdu _ tag）。数据域的所有块，除了最后一个外，使用 M _ apdu _ tag 发送。最后一个块使用 L _ apdu _ tag 发送。当接收到最后一个块，接收方把所有接收的数据域合并。这种机制对定义了两个标记值（M _ apdu _ tag、L _ apdu _ tag）的所有 APDU 和合法的，如图 F13 所示。

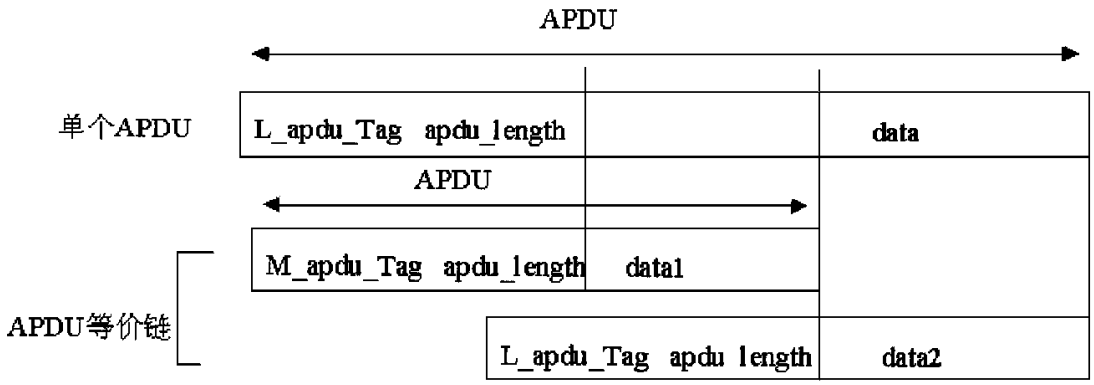


图 F13 链机制示意图

F6.3.3 在 SPDU 内传输 APDU

属于同一会话的整数个 APDU 可以在一个 SPDU 内传输。

F6.4 系统管理资源

F6.4.1 资源管理器

资源管理器是主机提供的一种资源。它只有一种类型，可以支持任何数量的会话。它控制所有应用的资源获取和供给。在模块和主机之间定义了一种对称的通信协议，以决定各自能提供的资源。主

机首先依次检查每个传输连接，决定传输连接上哪些资源可用。然后应用用它来查找所有可用的资源。

资源管理器周期性地检查资源变化情况，以便获得资源的整体情况。资源管理器由主机提供，不能由模块中的资源取代。任何试图从模块提供资源管理器的做法将被主机忽略。

F6.4.1.1 资源管理器协议

当一个模块插入主机或主机上电时，将产生一个或可能两个传输连接，为应用和/或资源提供者服务。应用或资源提供者首先所要做的是请求和资源管理器的资源建立连接。由于资源管理器没有会话的限制，这总是要发生的。资源管理器发送轮廓（Profile）查询指令给应用或资源提供者，后者以轮廓应答的方式列出所有的资源（如果有的话）。应用或资源提供者现在必须等待轮廓变更的结果。在等待轮廓变更的同时，不能和其他资源建立会话或者接受其他应用的会话，应返回“资源不存在”或“资源存在但不可用”等信息。

当主机查询了所有传输连接上的轮廓（Profile）并且接收到轮廓应答后，主机将列出所有可用资源的列表。当两个或多个资源级别、类型相同时，主机在列表中预留版本号最高的。当版本号也相同时，主机预留所有的资源并在接收到建立会话的请求时随机选择一个。一旦主机建立了它的资源列表，它将向所有的资源管理器的会话发送资源变更对象。那些想使用资源的应用可以使用轮廓查询对象查询主机获得资源列表。当第一次接收到轮廓变更提示时，应用或资源提供者可以使用轮廓查询来对主机查询，并以轮廓应答的方式接收到主机的可用资源列表。

在轮廓变更协议的这个操作之后，应用或资源提供者现在可以自由地建立或接收会话。它和资源管理器之间的会话可以接收主机随时进一步的轮廓变更通知。如果提供者想告知自己所提供的资源轮廓的变更，它将向主机发送一个轮廓变更通知。主机以轮廓查询作为应答，资源提供者进一步使用自己最新的资源列表作为应答。主机对此进行处理，如果结果导致主机自身的资源列表的任何变化，主机将向所有的有效的资源管理器有关的会话发送轮廓变更通知。应用就可以在想要的时候查询并接收到最新的资源列表。

F6.4.1.2 轮廓查询

轮廓查询对象要求接收方以轮廓应答对象的方式回答它的资源列表情况。见表 F17。

表 F17 轮廓查询对象编码

句法	比特数	类型

profile _ enq () {
profile _ enq _ tag
length _ field () =0
}
	24	uimsbf

F6.4.1.3 轮廓应答

轮廓应答作为对轮廓查询的响应，并列出了发送者所能够提供的资源列表。必须提供的资源的最小集合的资源标志在 F6.8 中列出。而且在本附录中给出了可选资源的定义。见表 F18。

表 F18 轮廓响应对象编码

句法	比特数	类型

profile _ reply () {
profile _ reply _ tag
length _ field () =N * 4
for (i=0; i<N; i++) {
resource _ identifier ()
}
}
	24	uimsbf

F6.4.1.4 轮廓变更

轮廓变更对象通知接收方某个资源发生了变化。模块通常用此来通知主机自己所提供资源的可用性发生了变化（不仅仅是一个资源在使用）。主机必要时改变自己的资源列表，而且在有变化时将对所有的传输连接发送轮廓变更对象。见表 F19。

表 F19 轮廓变更对象编码

句法	比特数	类型

profile _ changed () {
profile _ changed _ tag
length _ field () =0
}
	24	uimsbf

F6.4.2 应用信息

应用信息资源使得应用可以向主机给出自己的一套标准的信息。象资源管理器一样，它由主机提供并且没有会话限制。应用一旦结束了初始化的轮廓查询过程，它将建立和资源之间的会话连接。主机然后向应用发送应用信息查询对象，应用以应用信息对象返回合适的信息。

这个会话始终保持，以便主机随时可以通知应用在最上级的菜单进入点建立 MMI 会话。这是通过进入菜单对象实现的。当应用收到进入菜单对象时，它将建立 MMI 会话，并且显示自己的最上级菜单。主机应保证当它发送进入菜单对象时，会话能够建立。

F6.4.2.1 应用信息查询

见表 F20。

表 F20 应用信息查询对象编码

句法	比特数	类型

application _ info _ enq () {
application _ info _ enq _ tag
length _ field () =0
}
	24	uimsbf

F6.4.2.2 应用信息

见表 F21。

表 F21 应用信息对象编码

句法	比特数	类型
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

见表 F22。

表 F22 应用类型

application _ type	application _ type 值
CA	01
EPG	02
预留	其他值

application _ manufacturer

这个域的值从 CA 系统 ID 值中得到。

manufacturer _ code

这个域由各个厂商自定义。

menu _ string _ length

所有的应用有自己的用户菜单树，并且这是菜单的最上级，而在主机自己的菜单树中它可能是个子树。在此之后紧跟着一个描述菜单特性的字符序列。主机可以自由地决定自己的菜单树的结构，并使用应用类型域来对类似应用的菜单进入点分类。菜单可以仅仅是没有用户交互的简单显示，也可以是允许用户复杂操作的一整套负责的菜单屏幕。

text _ char

字符信息采用 GY/Z 174 的字符集和方法编码。

F6.4.2.3 进入菜单

见表 F23。

表 F23 进入菜单对象编码

句法	比特数	类型

enter _ menu () {
enter _ menu _ tag
length _ field () =0
}
	24	uimsbf

F6.4.3 条件接收支持

这个资源提供一整套对象以支持条件接收应用。象资源管理器一样，它由主机提供并没有会话的限制。所有应用结束初始化的应用信息状态后，可以和这个资源间建立会话连接。主机向应用发送 CA 信息查询对象，后者以包含合适信息的 CA 信息对象应答。这个会话将保持打开状态，这样 CA PMT、与 CA PMT 响应对象相关的协议就可以周期性地操作。

F6.4.3.1 CA 信息查询

见表 F24。

表 F24 CA 信息查询对象编码

句法	比特数	类型

ca _ info _ enq () {
ca _ info _ enq _ tag
length _ field () =0
}
	24	uimsbf

F6.4.3.2 CA 信息

见表 F25。

表 F25 CA 信息对象编码

句法	比特数	类型

ca _ info () {
ca _ info _ tag
length _ field ()
for (i=0; i<N; i++) {
CA _ system _ id
}
}
	24
	
	16
		uimsbf
		
		uimsbf

CA _ system _ id

它将列出应用支持的 CA 系统的 ID。

F6.4.3.3 选择需要解扰的业务

主机向一个或几个连接的 CA 应用发送 CA PMT，以指示用户选择了哪个基本流和如何找到 ECM

信息。每个 CA PMT 中包含选定节目的基本流的信息。如果用户选择了几个节目，这时需要发送几个 CA PMT 对象。

主机可以决定向所有连接的应用发送 CA PMT，还是仅仅向和具有选定的基本流 ES 中 CA _ descriptor 值一样的 CA _ system _ id 的应用发送。如果主机请求，每个主机将用 CA PMT 响应应答，允许主机选择进行解扰的模块。

F6.4.3.4 CA _ PMT

CA PMT 对象是由主机从 PSI 信息的 PMT 表中提取并向应用发送的表。这个表包含允许应用过滤 ECM 的信息，以及使得应用得到正确的加扰部分的 ECM 流的信息。见表 F26。

表 F26 CA PMT 对象编码

句法	比特数	类型
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) {		
ca _ pmt _ cmd _ id/ * 在节目层 * /	8	uimsbf
for (i=0; i<n; i++) {		
CA _ descriptor () / * 节目层的 CA 描述符 * /		
}		
}		
for (i=0; i<n; i++) {		
stream _ type		
reserved	8	uimsbf
elementary _ PID/ * 基本流的 PID * /	3	bslbf
reserved	13	uimsbf
ES _ info _ length		
if (ES _ info _ length ! =0) {	4	bslbf
ca _ pmt _ cmd _ id / * 在 ES 层 * /	12	uimsbf
for (i=0; i<n; i++) {		

表 F26 CA PMT 对象编码（完）

句法	比特数	类型
CA _ descriptor () / * 基本流中的 CA 描述符 * /	8	uimsbf
}		
}		
}		
}		

CA PMT 包括选定节目的所有 CA _ descriptors。如果选定了几个节目，主机将向应用发送几个 CA PMT 对象。CA PMT 只包含 CA _ descriptors。其他的描述由主机从 PMT 中去除。

在 current _ next _ indicator 后面的 CA _ descriptor 是节目级别的，对所有基本部件有效。基本流 ES 级别的 CA _ descriptor 只对 ES 有效。如果对一个 ES，CA _ descriptor 同时在节目级别、ES 级别存在，则只考虑 ES 级别的 CA _ descriptor。在以下情况，接收机向应用发送新的 CA PMT 或新的 CA PMT 的列表：

—用户选择另一个节目；

—通过一个“调谐”命令选择另一个业务；

—版本号发生变化；

—current _ next _ indicator 发生变化。

ca _ pmt _ list _ management

这个参数用来指示用户选择了单个节目（可以由一个或多个 ES 组成）还是多个节目。它可以采用表 F27 的值。

表 F27 ca _ pmt _ list _ management 取值

ca _ pmt _ list _ management	值
更多	00
第一个	01
最后一个	02
仅一个	03
加	04
更新	05
预留	其他值

当设置为“第一个”，表示是新的多于 1 个 CA PMT 对象的列表中的第一个。所有以前选择的节目将被新的列表的节目代替。“更多”表示 CA PMT 不是列表中的第一个，也不是最后一个。“最后一个”表示 CA PMT 对象是列表中的最后一个。“仅一个”表示列表只包含一个 CA PMT。“加”表示 CA PMT 需要加到现存的列表中，即用户选择了一个新的节目，但先前选择的节目仍然被选中。如果一个已经存在的节目接收到“加”，其操作和“更新”相同。“更新”表示列表中已经存在节目的 CA PMT，由于版本号或 current _ next _ indicator 变化需要重新发送。由应用负责来检查版本号的变化是否会引起 CA 操作的变化。由于所有的列表管理命令只在节目级别操作，现存节目的所有 ES 级别的变化必须

以“更新”命令方式通知节目级别，并且重新发送完成的 ES 流列表。

ca _ pmt _ cmd _ id

这个参数指示 CA PMT 对象要求应用做出何种响应。它可以取如表 F28 中的值。

表 F28 ca _ pmt _ cmd _ id 的取值

ca _ pmt _ cmd _ id	ca _ pmt _ cmd _ id 值
ok _ descrambling	01
ok _ mmi	02
query	03
Not _ selected	04
预留将来使用	其他值

当设置为 ok _ descrambling，主机不期望得到对 CA PMT 的任何应答，应用可以马上开始对节目解扰或者开始一个 MMI 对话过程。当设置为 ok _ mmi，应用可以开始 MMI 对话，但是在接收到一个新 ca _ pmt _ cmd _ id 设置为 ok _ descrambling 的 CA PMT 对象之前不能开始解扰。这种情况下，主机应保证 CA 应用可以打开 MMI 会话。当设置为 query，表示主机期望接收到一个 CA PMT 响应。这种情况下，应用在接收到一个 ca _ pmt _ cmd _ id 设置为 ok _ descrambling 或 ok _ mmi 的新的 CA PMT 对象之前不能开始解扰或 MMI 对话。当设置为 not _ selected 表示通知 CA 应用，主机不再需要这个应用去解扰服务，CA 应用将关闭所有的它打开的 MMI 对话。

F6.4.3.5 CA PMT 响应

这个对象总是由应用在接受到一个 ca _ pmt _ cmd _ id 设置为 query 的 CA PMT 对象后发送给主机。它也可以在接收到 ca _ pmt _ cmd _ id 设置为 ok _ mmi 的 CA PMT 后，由应用发送给主机，以表明 MMI 对话的结果（“可以解扰”如果用户已购买服务，或“不可解扰”（没有授权）如果用户没有购买）。见表 F29。

表 F29 CA PMT 响应对象编码

句法	比特数	类型
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/ * 在节目层 * /	7	uimsbf
else if (CA _ enable _ flag==0)		
reserved	7	bslbf

表 F29 CA PMT 响应对象编码（完）

for (i=0; i<n; i++) {		
reserved	3	blsbf
elementary _ PID/ * 基本流的 PID * /	13	uimsbf
CA _ enable _ flag	1	bslbf
if (CA _ enable _ flag==1)		
CA _ enable/ * 在基本流层 * /	7	uimsbf
else if (CA _ enable _ flag==0)		
reserved	7	bslbf
}		
}		

这个句法在节目级别可能包含 ca _ enable。对每个基本流 ES，在 ES 级别也可能包含 ca _ enable。

当两者同时存在时，只有 ES 级别的 ca _ enable 对那个基本流有效；

当两者都不存在时，主机不处理 ca _ pmt _ reply 对象。

CA _ enable 域表示应用是否能进行解扰操作。CA _ enable 的取值如表 F30。

表 F30 CA _ enable 的取值

CA _ enable	取值
可以解扰	01
满足条件时可解扰（购买对话）	02
满足条件时可解扰（技术对话）	03
不可解扰（因为没有授权）	71
不可解扰（因为技术原因）	73
预留将来使用	00—7F 间其他值

“可以解扰”表示节目不需要附加条件就可以解扰（如用户订单等），也不需要用户购买相应服务。“满足条件时可解扰（购买对话）”表示用户必须进入应用的购买对话购买服务，才能解扰（例如 PPV 按次付费节目）。“满足条件时可解扰（技术对话）”表示解扰前用户必须进入技术对话框（例如由于解扰能力有限要求用户选择较少的基本流）。“不可解扰（因为没有授权）”表示用户没有节目的授权或者不想购买这个节目。“不可解扰（因为技术原因）”表示应用因为技术原因不能解扰基本流（例如所有的解扰能力正在使用）。协议允许节目可以选择在节目级别或者 ES 级别解扰。当主机不支持不同的 ES 由不同模块解扰，则它把节目的 CA _ enable 值作为从节目的每个 ES 流的 CA _ enable 值中的最低值。这样保证任何流可以按照所要求的形式解扰。

F6.5 主机控制和信息资源

F6.5.1 主机控制

主机控制资源允许应用有限地控制主机的操作。在本版本中有两个主要的功能。第一个是重调谐主机到另一个业务。这将可能改变频率，甚至转向一个不同的卫星。前一个状态的上下文关系可能不

会被预留。第二个功能是暂时地用一个业务替代另一个。这可以持续一小段时间，例如插入广告，或者是长时间的替代整个节目。第一个功能由调谐对象支持。第二个功能由替代、取消替代对象支持。主机同时有一个请求释放对象请求应用关闭会话。只有主机才能提供主机控制资源，它同时只能支持一个会话。

F6.5.1.1 调谐

这允许应用把主机调谐到一个新业务。这个对象的参数包括网络 ID、原始网络 ID、TS 流 ID、业务 ID。当主机调谐到新的业务，它必须进入标准的 CA 支持对话，使用 CA _ PMT 对象使得新的业务可以被解扰。这样做有可能导致主机丢失当前的状态，主机在主机控制会话结束后并不重调谐到原先的业务，见表 F31。

表 F31 调谐对象编码

句法	比特数	类型
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
}		

F6.5.1.2 替代

替代和清除替代允许一个服务部件被同一个复接器的另一个服务部件暂时取代。应用分配一个 replacement _ ref 值，用来使用一个或多个替代对象来匹配，以实现清除替代对象。应用然后发送一个或多个替代对象，请求使用 replacement _ PID 中传输的部件来取代 replaced _ PID 中传输被取代的部件如视频、音频、文本、对白等。替代会马上发生。几个替代对象可以使用相同的 replacement _ ref，这种情况下当完全取代对象发送后，它们将全部被清除。主机应保持前一个服务的上下文关系，以便在发送完全取代对象后进行恢复。如果可能的话，当和主机控制资源之间的会话关闭时，前一个服务的上下文也将得到恢复。见表 F32。

表 F32 替代对象编码

句法	比特数	类型
replace () {		
replace _ tag	24	uimsbf
length _ field ()		
replacement _ ref	8	uimsbf
reserved	3	bslbf
replaced _ PID	13	uimsbf

表 F32 替代对象编码（完）

句法	比特数	类型
reserved	3	bslbf
replacement _ PID	13	uimsbf
}		

F6.5.1.3 清除替代

见表 F33。

表 F33 清除替代对象编码

句法	比特数	类型
clear _ replace () {		
clear _ replace _ tag	24	uimsbf
length _ field ()		
replacement _ ref	8	uimsbf
}		

F6.5.1.4 请求释放

当一个应用和主机控制资源之间有一个会话正在进行时，它将控制主机的行为。主机可以通过发送请求释放对象重新获得控制权。如果必要，应用将有一段时间可以发送清除替代对象并关闭会话。如果在规定的时间内会话未关闭，主机将强行关闭并在可能的情况下恢复先前的主机状态。见表 F34。

表 F34 请求释放对象编码

句法	比特数	类型
ask _ release () {		
ask _ release _ tag	24	uimsbf
length _ field () =0		
}		

F6.5.2 日期和时间

日期和时间资源由主机提供，可以支持无限多个会话。应用建立和这个资源的会话，然后通过日期—时间请求对象获得当前时间。如果 response _ interval 间隙为 0，则立即返回当前的日期—时间。如果 response _ interval 不等于 0，将首先返回当前的日期—时间，然后每间隔 response _ interval 周期则返回日期—时间。在可以兼容主机中，提供的时间是从 SI 信息的 TDT 中获得的。

F6.5.2.1 日期—时间查询

见表 F35。

表 F35 日期—时间查询对象编码

句法	比特数	类型
date _ time _ enq ( ) {		
date _ time _ enq _ tag	24	uimsbf
length _ field ( )		
response _ interval	8	uimsbf
}		

F6.5.2.2 日期—时间

见表 F36。

表 F36 日期—时间对象编码

句法	比特数	类型
date _ time ( ) {		
date _ time _ tag	24	uimsbf
length _ field ( ) = 5 或 7		
UTC _ time	40	bslbf
local _ offset/ * 可选 * /	16	tcimsbf
}		

UTC _ time

这载有 UTC 日期和时间（距离最近的整数秒数），采用 MJD 日期、小时、分钟、秒编码方式。

local _ offset

这是个可选域。如果存在的话，它对 UTC 时间和本地时间的差以有符号整数的方式进行编码，单位为分钟。本地时间=UTC 时间+local _ offset。当主机知道 local _ offset 的确切值时，它将提供 local _ offset，否则主机将忽略它。

F6.6 人机界面资源

F6.6.1 介绍

这里提供一类资源，它支持显示、以及用户通过键盘进行的互操作。这里定义了两个级别的互操作。低级别的 MMI 模式使得应用可以详细控制互操作的细节，包括接受用户的遥控按键、详细控制显示的布局和其他特性。高级别的 MMI 模式为应用提供一套高级别的语义，包括菜单、列表等。这种情况下主机决定显示的外观和感觉。在同一个会话中不能混合上述两种 MMI 模式。由主机决定是否能同时支持多于一个 MMI 会话。

高级别模式下显示对象传送的文本用 GY/Z 174 中的编码方法。低级别模式下显示对象传送的文本采用数字电视中有关字幕的编码方法。

F6.6.2 两种模式中使用的对象

F6.6.2.1 关闭 MMI

这个对象可以由主机或模块发送，允许用户或应用离开并把任何对话保留在高或低级别的 MMI

模式。编码见表 F37。

表 F37 关闭 MMI 对象编码

句法	比特数	类型
close _ mmi () {		
close _ mmi _ tag	24	uimsbf
length _ field ()		
close _ mmi _ cmd _ id	8	uimsbf
if (close _ mmi _ cmd _ id==delay) {		
delay	8	tcimsbf
}		
}		

close _ mmi _ cmd _ id

取值见 F38。

表 F38 close _ mmi _ cmd _ id 取值

close _ mmi _ cmd _ id	值
立即	00
延迟	01
预留	其他

表明显示是立刻返回到前面的业务，还是延迟一段时间以允许另一个 MMI 对话来取代它。

delay

如果另一个 MMI 会话未在同时启动，延迟（以秒为单位）一段时间后必须返回到当前的业务。当由应用发送时，主机应立刻关闭和应用相关的当前会话。如果 close _ mmi _ cmd _ id 为“立刻”，主机将同时立刻返回到先前的显示状态。如果 close _ mmi _ cmd _ id 为“延迟”，主机将保持 MMI 会话的最后状态，直到延迟时间到，或者另一个会话被启动时为止。这样做的目的是允许从一个应用到另一个应用的无缝 MMI 对话变化，而不会因为当前业务的加入而破坏视频显示。

如果应用关闭和 MMI 资源之间的会话，主机将把它理解为 close _ mmi _ cmd _ id 等于“立刻”的 close _ mmi 指令。当由主机发送时，应用将关闭当前的 MMI 会话。这样做是为了偏向于让主机来关闭和 MMI 资源相关的会话，以便应用可以更加方便地关闭。这种情况下延迟功能是没有必要的，主机将永远使用“立刻”关闭命令。

F6.6.2.2 显示控制

显示控制对象允许应用请求主机的图形/显示能力的三种工作模式：

—位图叠加视频；

—位图取代视频；

—基于字符的高级别 MMI 模式。

轮廓（Profile）通信允许应用获得以下信息：

—可寻址的显示坐标系统（典型地用来区分625、525行显示方式）；

—像素的幅型比；

—应用可使用的像素数/像素比特数；

—可用的字符集。

显示控制对象具有两个功能。一个是请求关于显示特性的信息。另一个是设置显示的模式。见表F39。

表F39 显示控制对象编码

句法	比特数	类型
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
}		
}		

显示控制命令按照表F40定义。

表F40 显示控制命令

display_control_cmd	值	命令描述
Set_mmi_mode	01	请求主机进入由MMI_mode字节指示的MMI模式
get_display_character_table_list	02	请求主机返回显示操作过程中可以支持的字符编码表的列表
get_input_character_table_list	03	请求主机返回输入操作过程中可以支持的字符编码表的列表
get_overlay_graphics_characteristics	04	请求主机在“图形叠加在视频上”的模式下的显示轮廓
get_full-screen_graphics_characteristics	05	请求主机在“图形取代视频上”的模式下的显示轮廓
预留	其他值	

人机界面模式取值按照表F41定义。

表 F41 人机界面模式取值

Mmi _ mode	值	描述
高级别	01	请求一个已打开的 MMI 会话。如果在主视频显示上实现，这将部分或全部使得正在显示的视频变得模糊
低级别图形叠加显示	02	请求打开低级别的图形 MMI 会话，叠加主视频显示（如果存在的话）
低级别图形全屏显示	03	请求打开低级别的图形 MMI 会话，取代主视频显示（或相互对立）
预留	其他值	

F6.6.2.3 显示响应

显示响应对象是作为主机显示系统对控制主机的应用所发出的显示控制对象的应答。见表 F42。

表 F42 显示响应对象编码

句法	比特数	类型
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

表 F42 显示响应对象编码（完）

句法	比特数	类型
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 {		
/* **所选人机模式的确认** */		
mmi _ mode	8	uimsbf
}		
}		
}		

display _ reply _ id

按照表 F43 **编码**。

表 F43 display _ reply _ id 编码

Display _ reply _ id	Id **值**
mmi _ mode _ ack	01
list _ display _ character _ tables	02
list _ input _ character _ tables	03
list _ graphic _ overlay _ characteristics	04
list _ full _ screen _ graphic _ characteristics	05
Unknown display _ control _ cmd	F0
unknown _ mmi _ mode	F1
unknown _ character _ table	F2
预留	**其他值**

display _ horizontal _ size，display _ vertical _ size

这 16 **比特整数描述了位图显示的最大可寻址坐标范围。所有的位置，从**（0，0）**到**（display _ horizontal _ size－1，display _ vertical _ size－1）**都可以被寻址。所有的接收机应支持最小** 720 **列。扫描行数最少比当时正在解码的视频图形垂直分辨率要大。因此当采用** 625/50 **视频工作时，至少支持** 576 **行；当采用** 525/60 **视频工作时，至少支持** 480 **行。**

aspect _ ratio _ information

这个 4 比特域和 GB/T 17975.1 中的 aspect _ ratio _ information 域编码方式相同，它允许从 display _ horizontal _ size，display _ vertical _ size 获得像素幅型比的信息。

graphics _ relation _ to _ video

这个 3 比特域表明了视频平面和图形平面之间关联程度。它将指明图形是否能按照预见的效果一样根据视频进行定位。广播者应注意到，任何使用降低分辨率播放的视频，在解码端可以通过与具体实现相关的位置处理进行扩展。在这些有限的条件下，图形可以在和视频不同的显示设备上显示。见表 F44。

表 F44　图形和视频的关系

图形和视频的关系	取值
视频与图像无关	000
预留	001-110
图形和视频坐标空间完全相同	111

multiple _ depths

当设置为 1 时，这个 1 比特域表示显示是否可以支持不同显示像素深度的组合。(如每个显示区使用特定的显示像素深度)。当设置为 0 时，这个域表示显示系统对不同显示像素深度的混合做出限制。这种情况下，所有的区域使用单个像素深度。

display _ bytes

这个 12 比特整数乘以 256 后，表示显示内存可用的字节数。

composition _ buffer _ bytes

这个 12 比特整数乘以 256 后，表示组合缓冲区可用的字节数。

object _ cache _ bytes

这个 8 比特整数乘以 4 096 后，表示对象高速缓冲区可用的字节数。

number _ pixel _ depths

这个 4 比特整数表示显示可以支持的不同的显示像素深度的数量。

display _ depth

这个 3 比特域表示显示像素深度。

pixels _ per _ byte

这个 3 比特整数给出在当前显示深度下，可以组合进一个字节的像素个数。取值为 0 表示 8 比特深度显示的特例。

region _ overhead

这个 8 比特整数乘以 16 后，给出了当在此显示深度下引入新的显示区域后，可显示的像素的减少数量。接收机在下列公式成立的条件下应能够实现应用提出的显示请求：

显示字节数 display _ bytes＞＝正在使用的字节数 bytes _ used

这里已使用的字节数的计算方法如下：

```
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
```

在 GY/Z 174 **中，字符域的字符表选择（如果和缺省的不同）由字符域的初始字节表示。字符表格字节是按** GY/Z 174 **中定义的字符表格字节的无序排列。**

F6.6.3 低级别 MMI **键盘对象**

这个对象只在低级别 MMI **模式下使用。**

F6.6.3.1 键盘控制

应用假定主机具有键盘，用户使用键盘进行人机对话。这个键盘可以是虚拟的，用户可以输入命令或利用其他方式（如声音），主机把用户的操作解释成按键操作，并提供给应用。见表 F45。

表 F45 **键盘控制对象编码**

句法	比特数	类型
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++) {		
/* **接受的按键列表** */		
key_code	8	uimsbf
}		
}		
if (keypad_control_cmd == ignore_selected_keypress) {		
for (i=0; i<keypad_control_length−1; i++) {		
/* **忽略的按键列表** */		
key_code	8	uimsbf
}		
}		
if (keypad_control_cmd == reject_keypress) {		
key_code/* **拒绝的按键** */	8	uimsbf
}		
}		

键盘控制对象和下面描述的按键对象允许应用截取按键。键盘控制对象允许应用把按键提交给其他应用。它们是通过按键对象发送的。应用也可以停止接收按键，或者拒绝自己不清楚或不感兴趣的按键。键盘控制命令见表 F46。在 F6.6.3.3 中定义了可以被截取的按键。

表 F46　键盘控制命令取值

Keypad _control _cmd	Cmd 取值
intercept _all _keypresses	01
ignore _all _keypresses	02
intercept _selected _keypress	03
ignore _selected _keypress	04
reject _keypress	05
Reserved	其他值

F6.6.3.2　按键

见表 F47。

表 F47　按键对象编码

句法	比特数	类型
keypress () {		
keypress _tag	24	uimsbf
length _field () =1		
key _code	8	uimsbf
}		

F6.6.3.3　按键编码表

这些是在主机/应用接口使用的按键编码。对应的键不一定必须在键盘上存在，但主机必须知道如何把对应的用户操作翻译成响应的按键编码。对所有按键编码的支持是必须的要求。按键编码值（16进制）如表 F48 所示。

表 F48　按键编码表

按键编码	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
含义	0	1	2	3	4	5	6	7	8	9	菜单	Esc	⇒	⇐	⇑	⇓	BS	RC

其他值（从 0x12 到 0xFF）为预留值。

F6.6.4　低级别 MMI 显示对象

低级别 MMI 显示能力是基于字幕机制实现的。

F6.6.4.1　字幕数据的传送

字幕段采用下面描述的 CI 多密接口 APDU 的形式进行封装。当每个完整的显示字幕从应用传送到主机时，应用应发送一个场景结束标记 APDU。这个 APDU 将用来划清不同显示字幕的界限，也为

应用提供对解码过程的暂时控制。

F6.6.4.2 段封装

字幕段按照CI接口APDU的方式封装。为了使长的段能在多个APDU上分块，将使用更多/最后一个的格式。见表F49。

表F49 字幕段对象编码

句法	比特数	类型
subtitle _ segment () {		
subtitle _ segment _ tag	24	Uimsbf
length _ field ()		
Subtitling _ segment ()		
}		

F6.6.4.3 显示系统信息

显示系统信息向应用告警，提示应用注意需要引起注意的环境和场景。编码和取值见表F50和F51。

表F50 显示消息对象编码

句法	比特数	类型
display _ message () {		
display _ message _ tag	24	uimsbf
length _ field ()		
display _ message _ id	8	uimsbf
}		

表F51 显示消息取值

display message _ id	取值	何时使用
显示正确	00	作为主机做出确认的可选输出，对任何低或高级别MMI对象确认。
显示错误	01	显示系统检测到一个错误
显示内存不够	02	主机的组合缓冲区、像素缓冲区、对象高速缓存用尽的错误指示
字幕语法错误	03	指示主机无法解释字幕段的错误
引用了未定义的区域	04	指示主机发现了一个未定义region _ id的区域被引用的错误
引用了未定义的CLUT	05	指示主机发现了一个未定义CLUT _ id的CLUT被引用的错误
引用了未定义对象	06	指示主机发现了一个对未定义的object _ id引用的错误

表 F51 显示消息取值（完）

display message _ id	取值	何时使用
对象和区域不兼容	07	表示像素深度或对象大小和使用的区域不兼容
引用了未知的字符	08	指示发现了一个和选定的字符表不兼容的字符编码
显示特性发生变化	09	这个消息表示在应用最后检测之后，显示的某些特性发生了变化。例如用户对 TV 从 16：9 重新设置为 4：3，或根据节目类型的改变做出变化（如解码的视频格式的变化）。
预留	其他值	

F6.6.4.4 时间控制

当广播字幕时，解码和连续字幕信息显示的定时由封装字幕段的 PES 头的 PTS 来控制。当在 CI 接口上使用时，一整套 APDU 可提供时间控制。

scene _ end _ mark

见表 F52。

表 F52 场景结束标记对象编码

句法	比特数	类型
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
}		

当一屏（或多屏）的字幕段发送给解码器之后，应用应发送一个场景结束标记 APDU。场景结束标记告知接收机数据集合的结束，并告诉接收机在结束接收后该做什么。

decoder _ continue _ flag

这个 1 比特标志设为‘1’时，告诉接收机继续（从这个 MMI 会话）解码字幕数据。当设为‘0’时，解码过程将停止（在场景结束标记中其他指令的操作完成后）。

scene _ reveal _ flag

这 1 比特标志设为‘1’时，告诉接收机显示先前的组成段的页。即显示应从当前有效的组成段的页变更到最近刚解码的组成段的页。当标志设为‘0’，组成段的页的实现一直推迟到含有匹配的 scene _ tag 的场景显示对象被发送为止。

send _ scene _ done

这个 1 比特标志设为 '1' 时，告诉接收机发送一个场景完成的 APDU 给应用。

scene _ tag

这个 4 比特整数由应用设置。它的值按照每个场景结束标记以模 16 的方式递增。

scene _ done _ message

见表 F53。

表 F53 场景完成消息编码

句法	比特数	类型
scene _ done _ massage () {		
scene _ done _ message	24	uimsbf
length _ field ()		
decoder _ continue _ flag	1	bslbf
scene _ reveal _ flag	1	bslbf
reserved	2	bslbf
scene _ tag	4	uimsbf
}		

当接收机结束一个场景结束标志后的显示消息的全部解码后，接收机将发送场景完成消息。

decoder _ continue _ flag，scene _ reveal _ flag

这些 1 比特标志复制在导致消息被发送的场景结束标记中相同的标记的状态。

scene _ tag

这些 4 比特整数复制在导致消息被发送的场景结束标记中相同的整数的状态。

scene _ control

编码见表 F54。

表 F54 场景控制编码

句法	比特数	类型
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
}		

只有接收机发送对应显示的场景完成消息后，应用才能发送场景控制 APDU。

decoder _ continue _ flag

这个 1 比特标志设为 '1' 时，指示接收机继续解码字幕数据。在场景结束标志中，只有接收机 decoder _ continue _ flag 设置为 '0' 时，这个标志才有用。

scene _ reveal _ flag

这个 1 比特标志设为 '1' 时，指示接收机继续完成这个场景的对白数据块的新页。在场景结束标志中，只有接收机 scene _ reveal _ flag 设置为 '0' 时，这个标志才有用。

scene _ tag

这个 4 比特整数表示正在被操作的场景结束标志。它的值对每个场景控制按照模 16 的方式递增。

F6.6.4.5 对象下载

字幕下载 APDU 和字幕段的 APDU 格式完全相同。对这个 APDU 的限制是它们只能载有字幕对象数据段（而字幕段 APDU 可以载有任何字幕段）。这个对象的实现有几个可选项。而且提供给应用的功能和下面的内容是等价的：

—由 APDU 承载的字幕对象数据段存储在对象高速缓存中；

—当一个区域引用高速缓存中的对象时（使用合适的对象 ID、对象提供者的标志），字幕段将从高速缓存中被读出并送到接收机。

最差的情况下，接收机的响应速度等同于其需要数据段时才从 CI 接口传送数据段的速度。然而应用的处理负担将被降低，接收机的处理负担也将得到降低。

字幕可以识别的对象类型都可以存储在高速缓存中。然而如果对象类型是字符或字符串，获得每个字符的字型的方法在本指导性技术文件中未规范。当字幕段 APDU 可以载有对象数据段时，就可以发送字幕下载 APDU。当打开 MMI 会话时，总是假定在高速缓存中没有数据。当 MMI 会话关闭时，缓存中的数据将被消除。字幕下载编码见表 F55。高速（Flush）下载编码见表 F56。下载响应编码见表 F57。

表 F55 字幕下载编码

句法	比特数	类型

Subtitle _ download () {
Subtitle _ download _ tag
Length _ field ()
Subtitling _ segment ()
}
	24	uimsbf

表 F56 Flush 下载编码

句法	比特数	类型

flush _ download () {
flush _ download _ tag
length _ field ()
}
	24	uimsbf

这要求接收机的字幕对象高速缓存被清除。

表 F57 下载响应编码

句法	比特数	类型
Download _ reply () {		
Download _ reply _ tag	24	uimsbf
length _ field ()		
object _ id	16	uimsbf
download _ reply _ id	8	uimsbf
}		

下载响应对象允许主机支持对象下载的问题。当下载成功时，将不需要此响应。

object _ id

表示产生消息的已下载对象的 ID。当消息报告此段不是一个数据段时，object _ id 应设为 0xFFFF。见表 F58。

表 F58 下载响应标识

Download _ reply _ id	取值
下载成功	00
不是对象数据段	01
内存耗尽	02
预留	其他值

F6.6.4.6 从 CI 接口寻址时的字幕解码器模型

这里对处理速度没有任何附加的要求。然而如果可能的话，接口可以利用更快的处理速度。当工作在叠加模式下时，希望字幕解码器可提供和数字电视字幕的有关标准中定义相同的像素缓冲区内存、组合缓冲区内存。然而当从 CI 接口寻址时，将为应用提供关于显示系统的足够信息，以保证不会发生类似不在预计之中的颜色量化现象。

全屏幕图形模式未定义。当支持这种模式时，需要以下主机资源：

最小的像素缓冲区内存 207360 字节（原则上可提供 720 x 576 x 16 色）

最小 12K 字节的缓冲区内存

对对象高速缓存 RAM 没有必须的最小要求。当显示轮廓表明没有对象高速缓存 RAM 时，应用将不能使用对象下载功能。

——对象高速缓存的使用

位图对象下载包括 object _ id 以及在区域组合段中设置为 0x01 的 object _ provider _ flag（表明对象提供者驻留在主机中的 ROM 中）。解码器最坏的速度情况和当需要时每个段才从 CI 接口传送过来的情况相同。然而使用高速缓存，可以降低应用的处理负担、解码器的处理负担。因此这个技术提供了一个提高系统性能的机会。提供对象高速缓存是建议性的要求，而不是强制性要求。

——页标志

CI接口对字幕的使用，要求解码器就象字幕段是插入在字幕解码器已编码的数据缓冲区之前的“字

幕处理”模块的输入位置一样。因此这个插入点是在过滤器之后，字幕段可以被解码器忽略，也不需要由应用设置。

——其他字幕业务

解码器被假定为只有单个字幕解码器。如果解码器已经在解码一个字幕流，当 MMI 会话启动解码器时，解码器在 MMI 会话期间将会暂时挂起广播数据的解码。当会话被关闭后，如果可能的话，解码器将返回到先前的解码任务。

F6.6.5 高级别 MMI 对象

下面的对象仅仅在高级别 MMI 模式下使用。高级别 MMI 模式对象定义了所需要的会话，允许主机决定显示的类型、格式。

应用可以使用文本中的控制特性，但这些可能被主机忽视。这些控制特性可以由主机用于帮助菜单的显示、展开。文本的任何内容将根据当前的字符集来解释。主机信息的表达并不受所使用的显示设备的限制。

F6.6.5.1 文本

文本对象用于在屏幕上显示一块文本。它在高级别 MMI 模式中用作高级别对象的一部分。见表 F59。

表 F59 文本对象编码

句法	比特数	类型
text () {		
text _ tag	24	uimsbf
length _ field ()		
for (i=0; i<length; i++)		
text _ char	8	uimsbf
}		

文本长度等于 0 的文本对象将被解释为空对象，将不会显示任何信息。

text _ char

文本信息采用 GY/Z 174 中规定的字符集和编码方法进行编码。应用发送的文本可以包含相应的 GY/Z 174 中定义控制字符，以指示文本的显示方式。对这些控制字符的解释由主机实现。

F6.6.5.2 查询

查询和应答允许应用请求用户输入。例如一个请求用户输入 PIN 码的请求。对查询的响应由主机以应答对象的方式回应。查询对象允许应用来明确规定用户输入是否应由主机反馈给用户。例如在输入 PIN 码时，用户输入的数字不应该显示出来。见表 F60。

表 F60 查询对象编码

句法	比特数	类型
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：设置为 1 时，表示用户输入不应显示。主机可以选择用字符替代（如星号等）。

answer _ text _ length：期望的应答的长度。如果应用不知道，将设为 0xFF。

text _ char

文本信息采用 GY/Z 174 中定义的字符集、编码方法进行编码。

查询的例子见图 F14。

请输入 PIN 码 /* 查询文本 */
* * * * /* 掩码显示 */

图 F14 查询例子图例

F6.6.5.3 应答

这个对象和查询对象一起用来管理用户输入。见表 F61。

表 F61 应答对象编码

句法	比特数	类型
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
}		
}		

当应用希望从用户处接收信息时，它将发送查询对象。主机将返回应答对象（在把按键码翻译成

字符编码之后)。在应答对象中的文本将使用和相关查询对象相同的编码方法。在应答对象中的文本将使用和相关的查询对象相同的方法来选择字符集(在GY/Z 174中定义)。通常使用answ _id域来表示从用户处接收到的响应的类型。answ _id的值按表F62进行编码。

表F62 answ _id取值

answ _id	取值
取消	00
应答	01
预留	其他值

取值“应答”表示目标包含有用户输入(也可以是0长度)。取值“取消”表示用户想放弃这个对话。

F6.6.5.4 菜单

这个对象和菜单应答对象一起用来在高级别MMI模式下管理菜单。见表F63。

表F63 菜单对象编码

句法	比特数	类型
menu () {		
menu _tag	24	uimsbf
length _field ()		
choice _nb	8	uimsbf
TEXT () /* 标题文本 */		
TEXT () /* 子标题文本 */		
TEXT () /* 结束行文本 */		
for (i=0; i<choice _nb; i++) /* 当 choice _nb! =FF 时 */		
TEXT ()		
}		

菜单一般由一个标题、一个子标题、几个选择项和一个结束行组成。可以使用with text _length=0的对象(如没有子标题、没有结束行文本)。Choice _nb=FF意味着这个域未载有选择项的个数信息。主机显示文本、菜单框、选择相应选项的实现方法和具体厂家有关。例如主机可以自由地在几个页上显示选项,并且自己管理向上、向下翻页功能。厂家可以自由地定义用户如何选择这些选项(如数字键盘、箭头、带颜色的键、声音等)。

F6.6.5.5 菜单应答

这个对象和菜单对象一起使用返回用户选择,并表明用户已完成相应对象。见表F64。

表 F64 菜单应答对象编码

句法	比特数	类型
menu _ answ () {		
menu _ answ _ tag	24	uimsbf
length _ field () = 1		
choice _ ref	8	uimsbf
}		

choice _ ref：

用户选择的选项的编号。如果在此对象之前有一个菜单对象，choice _ ref＝01 **对应于应用在那个对象中提供的第一个选项（菜单对象中结束行文本后的第一个选项文本），**choice _ ref＝02 **对应于应用在那个对象中提供的第二个选项，**choice _ ref＝00 **表示用户没有做出选择而是取消了先前的菜单或对象列表。下面是菜单的一个例子：**

想买此节目吗？/＊**标题**＊/

《**生死抉择**》**价格**：12.00 **元**/＊**子标题**＊/

1/**是**/＊**选择** 1＊/

2/**否**/＊**选择** 2＊/

〈**账户余额**：47.50 **元**〉/＊**结束行**＊/

〈**前三分钟免费**〉

F6.6.5.6 **列表**

这个对象用户发送一个用于显示的事件的列表（例如在授权的过程中）。它和菜单对象的句法相同，一般和菜单应答对象一起使用。见表 F65。

表 F65 列表对象编码

句法	比特数	类型
list () {		
list _ tag	24	uimsbf
length _ field ()		
item _ nb	8	uimsbf
TEXT () /＊**标题文字**＊/		
TEXT () /＊**子标题文字**＊/		
TEXT () /＊**结束行文字**＊/		
for (i＝0；i＜item _ nb；i＋＋) /＊**当** item _ nb！＝FF **时**＊/		
TEXT ()		
}		

列表由一个标题、一个子标题、几个选项和一个结束行组成。文本长度 text _ length＝0 **的文本对象也可以使用（如当没有子标题或没有结束行文本在使用时）。**Item _ nb＝FF **表示这个域未包含选项的**

个数信息。

主机显示标题、子标题、结束行文本、选项的方式是和厂商有关的。例如，主机可以自由选择在几个页面上显示选项、管理自己的向上、向下翻页功能。

F6.7 通信资源

F6.7.1 低速通信资源级别

F6.7.1.1 简介

主机的一部分是与低速通信资源级别的接口。它将提供在例如电话线、有线电缆回传通道上的双向通信功能。它可以被用于条件接收，也可以和交互性业务一起使用。这个资源类别用通常的模式定义，以便不同的通信技术可以使用共同的方式实现。

在级别内部定义了资源类型，以使用通用对象集来支持与特定类型设备的通信。这个模式是同时双向（全双工）方式，在此之上可以传输任意的数据。在应用和主机之间的每个方向上将使用流量控制。为了降低对主机、应用的缓冲区大小的要求，数据将被分割成块进行传输。流量控制协议同时降低了主机、应用中缓冲区的数量的要求。

下面定义的是主机和应用之间的低速通信功能使用的 APDU。

F6.7.1.2 需求

a）每个方向上至少具有支持 254 字节的缓冲区。如果需要，应用可以选择低于此值的缓冲区大小；

b）流量控制在主机和应用之间实现。当第一个缓冲区数据正在发送时，主机应使用第二个缓冲区来传送数据，但第三个必须等待第一个缓冲区的数据发送完毕。通信接口输入的数据的操作协议是类似的。主机负责完成与外部通信连接之间建立的连接的流量控制，并和应用/主机之间的流量控制协同工作。

F6.7.1.3 支持低速通信资源的对象

这里一共定义了 Comms Cmd、Comms Reply、Comms Send、Comms Rcv 四种对象。Comms Cmd 由应用发送，以允许在通信资源上执行几个管理操作；Comms Reply 由主机发出以对 Comms Cmd 对象进行确认。它也可作为对 Comms Send 的确认，以对通信资源的输出进行流量控制；Comms Send 是发送给通信资源的缓冲区中的数据；Comms Rcv 是从通信资源接收来的缓冲区的数据，即从线路接收的数据。缓冲区大小、以及 Comms Cmd 对象早些时候设置的超时值对接收缓冲区有效。

为使用特定的通信资源，必须使用标准的会话建立机制来建立会话。在此会话上可以使用会话对象进行通信。

F6.7.1.4 Comms Cmd

Comms Cmd 对象编码见表 F66。连接描述子对象编码见表 F67。Comms _ command _ id 取值见表 F68。Connection _ descriptor _ type 取值见表 F69。

表 F66 Comms Cmd 对象编码

句法	比特数	类型
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
}		
}		

表 F67 连接描述子对象编码

句法	比特数	类型
connection _ descriptor () {		
connection _ descriptor _ tag	24	uimsbf
length _ field ()		
connection _ descriptor _ type	8	uimsbf
if (connection _ descriptor _ type ==		
SI _ Telephone _ Descriptor) {		
telephone _ descriptor ()		
}		
if (connection _ descriptor _ type ==		
Cable _ Return _ Channel _ Descriptor) {		
channel _ id	8	uimsbf
}		
}		

表 F68 Comms _ command _ id 取值

Comms _ command _ id	id 取值
Connect _ on _ Channel	01
Disconnect _ on _ Channel	02
Set _ Params	03
Enquire _ Status	04
Get _ Next _ Buffer	05
预留	其他值

表 F69 Connection _ descriptor _ type 取值

Connection _ descriptor _ type	类型取值
SI _ Telephone _ Descriptor	01
Cable _ Return _ Channel _ Descriptor	02
预留	其他值

Connect _ on _ Channel 建立通信资源上的通信连接。Connection _ descriptor 包含建立连接的必要信息，如使用调制解调器时的拨号电话号码等。Retry _ count 指可以重试的次数，超时（以秒为单位）表示如果在特定时间内未接收到连接状态的应答就可以放弃该连接。当超时值为 0 时，表示无限等待下去。

Disconnect _ on _ Channel 结束与通信资源的连接。Set Params 带有两个参数。第一个是以字节为单位给出最大缓冲区大小的 8 比特值，其最大值为 254、最小值为 1。第二个参数是个 8 比特值，以10 ms 为单位给出输入超时值。如果在超时界限到来之前，当前缓冲区接收到 1 个或多个字节，而且没有收到进一步的字节，则缓冲区将作为 Comms Rcv 对象传送给应用。如果超时界限到来前，缓冲区到达了缓冲区大小参数设定的上限，这个缓冲区将被立即返回。

Enquire _ Status 没有参数，并产生一个表明当前通信连接状态的 Comms Reply 对象。

Get _ Next _ Buffer 在接收端对流量控制协议进行操作。当通道上成功建立一个连接后，应用发出 comms _ phase _ id 为 0 的 Get _ Next _ Buffer 对象。对第一个对象 comms _ phase _ id 必须为 0，主机也要求它为 0。如果愿意的话，应用可以发送 comms _ phase _ id 为 1 的 Get _ Next _ Buffer 的对象。直到从通道中以接收 Comms Rcv 对象的方式接收到 1 个缓冲区数据位置，它可以不再发送数据。当接收到缓冲区数据后，又可以发送 comms _ phase _ id 为 0 的 Get _ Next _ Buffer 对象。这样 comms _ phase _ id 以 0、1、0、1…值轮流使用。通过这种手段，应用可以控制主机发送数据的速率，这样取值也允许应用清楚地识别已发送的是何种数据。

发出 Get _ Next _ Buffer 对象应马上得到 Comms Reply 对象的确认，稍后应以 Comms Rcv 对象的形式接收到缓冲区的数据。

F6.7.1.5 Comms Reply

Comms Reply 对象编码见表 F70。comms _ reply _ id 取值见表 F71。

表 F70 Comms Reply 对象编码

句法	比特数	类型
comms _ reply () {		
comms _ reply _ tag	24	uimsbf
length _ field ()		
comms _ reply _ id	8	uimsbf
return _ value	8	uimsbf
}		

表 F71 comms _ reply _ id 取值

comms _ reply _ id	Id 取值
Connect _ Ack	01
Disconnect _ Ack	02
Set _ Params _ Ack	03
Status _ Reply	04
Get _ Next _ Buffer _ Ack	05
Send _ Ack	06
Reserved	其他值

通常正的返回值表示 OK，负的返回值表示错误。0 是标准的 OK 返回值，01 表示未知的错误原因。进一步的错误值还需要定义。Status _ Reply 的返回值为 0 时表示未连接，为 1 是表示已连接。Send _ Ack 的返回值指示哪个缓冲区已被成功发送，对所有的发送的缓冲区数据，它取值为 0 或 1。0 表示在一个成功的“连接”Comms Cmd 对象之后发送的第一个缓冲区数据。后面的缓冲区数据轮流使用 1、0、1、0。当使用 0 状态的缓冲区数据发送后，得到一个返回值为 0 的 Send _ Ack 的应答，就可以发送下一个数据。同样当使用 1 状态的缓冲区数据发送后，得到一个返回值为 1 的 Send _ Ack 的应答，就可以发送下一个数据。这个机制控制应用与通信的数据传送流量控制，并且只要求主机提供两个缓冲区—当前正在发送的缓冲区、下一个使用的缓冲区。

在没有错误发生的情况下，这个对象可以主动发送。唯一的错误是连接拆除，这种情况下主机发送 comms _ reply _ id 为 0 的 comms _ reply 对象（表示连接已拆除）。

F6.7.1.6 Comms Send

见表 F72。

表 F72 Comms Send 对象编码

句法	比特数	类型
comms _ send () {		
comms _ send _ tag	24	uimsbf
length _ field ()		
comms _ phase _ id	8	uimsbf
for (i=0; i<n; i++) {		
message _ byte	8	uimsbf
}		
}		

Comms _ phase _ id 取值为 0 或 1。在连接 Comms Cmd 对象之后的第一个 Comms Send 对象必须设置 comms _ phase _ id 为 0。后续的 Comms Send 对象必须轮流使用 1、0、1、0 序列。主机将对此做出检查，在此序列发生错误的情况下以 Comms Reply 对象的形式返回 Send _ Ack 错误信息。这将允许应用清楚地识别哪个缓冲区数据已经被确认。采用 1、0 两个状态而不是发送—确认—发送—确认机制，将使得通信缓冲区可以连续地添满数据进行发送，以便可以在最大速率下工作。

消息的最大字节数为 254。

F6.7.1.7 Comms Rcv

见表 F73。

表 F73 Comms Rcv 对象编码

句法	比特数	类型
comms _ rcv () {		
comms _ rcv _ tag	24	uimsbf
length _ field ()		
comms _ phase _ id	8	uimsbf
for (i=0; i<n; i++) {		
message _ byte	8	uimsbf
}		
}		

Comms _ phase _ id 表示数据所属于的 Get _ Next _ Buffer 周期的状态。

F6.8 资源标志和应用对象标记

F6.8.1 资源标志

下面定义了本附录中的公共资源的标志。见表 F74。

表 F74 资源标志值

资源	级别	类型	版本	资源标志
资源管理器	1	1	1	00010041
应用信息	2	1	1	00020041
条件接收支持	3	1	1	00030041
主机控制	32	1	1	00200041
日期—时间	36	1	1	00240041
MMI	64	1	1	00400041
低速通信	96	见 F6.8.1.1	1	0060xxx1
预留	其他值	其他值	其他值	其他值

低速通信资源类型

低速通信资源类型值包括两个域。第一个域使用资源类型的 0、1 比特，表示设备号。对特定设备来说，可以有多个实例，设备号域将标志使用哪个实例。第二个域使用 2—9 比特，表示合适的设备类型。对特定设备如调制解调器，它又可以划分成子域。见图 F15。

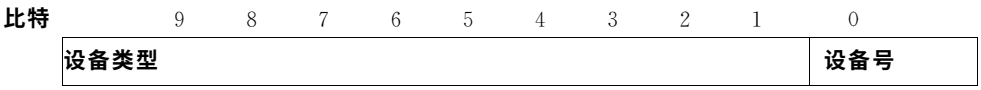


图 F15 低速通信资源类型结构

设备类型域按表 F75 编码。

表 F75 设备类型域编码

描述	取值
调制解调器	00—3F
串口	40—4F
电缆回传通道	50
预留	51—FF

调制解调器的资源类型域分为三个子域。设备号和上面的描述相同，但设备类型分为数据处理需要的域、按照 ITU-T V 系列规范的速度进行编码的调制解调器类型。见图 F16。

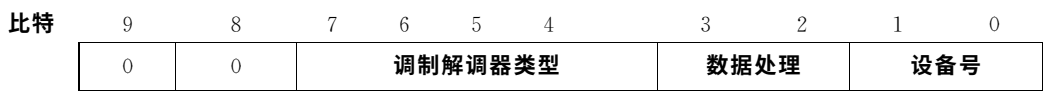


图 F16 调制解调器的资源类型结构

数据处理域按表 F76 编码。

表 F76 数据处理域编码

描述	取值
协商	00
非 V.42bis	01
没有数据处理	10
预留	11

“协商”表示如果和远端协商成功，则使用 V.42 误码纠正和 V.42bis 数据压缩技术。“非 V.42bis”表示使用 V.42 错误纠正方法，但不使用 V.42bis 数据压缩技术。

调制解调器类型域按表 F77 编码。

表 F77 调制解调器类型域编码

描述	取值
预留	0000
预留	0001
V.21（300 bps）	0010
预留	0011
V.22（1 200 bps）	0100
V.22bis（2 400 bps）	0101
V.23（1 200/75 bps）	0110
预留	0111
V.32（9 600/4 800 bps）	1000
V.32bis（14.4 kbps）	1001
V.34（28.8 kbps）	1010
预留	1011—1101
V.27ter	1110
V.29	1111

如果主机使用的特定的调制解调器提供多个 V. 系列速度，则主机应把所有的速度作为不同的资源类型提供。如果一个特定的资源类型正在使用，则那个调制解调器所提供的其他类型的资源也处于忙的状态。

F6.8.2 应用对象标记

Apdu _ tag 的编码遵循 ASN.1 规则。每个 apdu _ tag 采用 3 个字节编码。在每个 apdu _ tag 的 24 比特编码中，10 比特由 ASN.1 规定并按照下图编码，并且只对原始的标记进行编码。见图 F17。

字节 1		字节 2		字节 3	
b24	b17	b16	b9	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	

图 F17 原始标记编码

应用对象标记取值见表F78。

表F78 应用对象标记取值表

apdu _ tag	标记值（HEX）	资源	方向（主机<——>应用）
T profile _ enq	9F 80 10	资源管理	<——>
T profile	9F 80 11	资源管理	<——>
T profile _ change	9F 80 12	资源管理	<——>
T application _ info _ enq	9F 80 20	应用信息	——>
T application _ info	9F 80 21	应用信息	<——
T enter _ menu	9F 80 22	应用信息	——>
T ca _ info _ enq	9F 80 30	CA 支持	——>
T ca _ info	9F 80 31	CA 支持	<——
T ca _ pmt	9F 80 32	CA 支持	——>
T ca _ pmt _ reply	9F 80 33	CA 支持	<——
T tune	9F 84 00	主机控制	<——
T replace	9F 84 01	主机控制	<——
T clear _ replace	9F 84 02	主机控制	<——
T ask _ release	9F 84 03	主机控制	——>
T date _ time _ enq	9F 84 40	日期—时间	<——
T date _ time	9F 84 41	日期—时间	——>
T close _ mmi	9F 88 00	MMI	——>
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	——>

应用对象标记取值见表F79。

表F79 应用对象标记取值表

apdu _ tag	标记值（HEX）	资源	方向（主机<——>应用）
T enq	9F 88 07	MMI	<——
T answ	9F 88 08	MMI	——>
T menu _ last	9F 88 09	MMI	<——
T menu _ more	9F 88 0A	MMI	<——
T menu _ answ	9F 88 0B	MMI	——>

表 F79 应用对象标记取值表（完）

apdu _ tag	标记值（HEX）	资源	方向（主机<——>应用）
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	低速通信	<———
T connection _ descriptor	9F 8C 01	低速通信	<———
T comms _ reply	9F 8C 02	低速通信	———>
T comms _ send _ last	9F 8C 03	低速通信	<———
T comms _ send _ more	9F 8C 04	低速通信	<———
T comms _ rcv _ last	9F 8C 05	低速通信	———>
T comms _ rcv _ more	9F 8C 06	低速通信	———>

F7 基于 PC 卡的物理层实现

F7.1 通用描述

模块和主机之间的接口的外形、机械性能是 PC 卡标准，第 2 卷（电气特性规范，1995 年 2 月，PCMCIA 国际协会）、PC 卡标准，第 3 卷（物理特性规范，1995 年 2 月，PCMCIA 国际协会）、PC 卡标准，第 4 卷（Metaformat 规范，1995 年 2 月，PCMCIA 国际协会）的变形。图 18 给出了典型的模块结构。

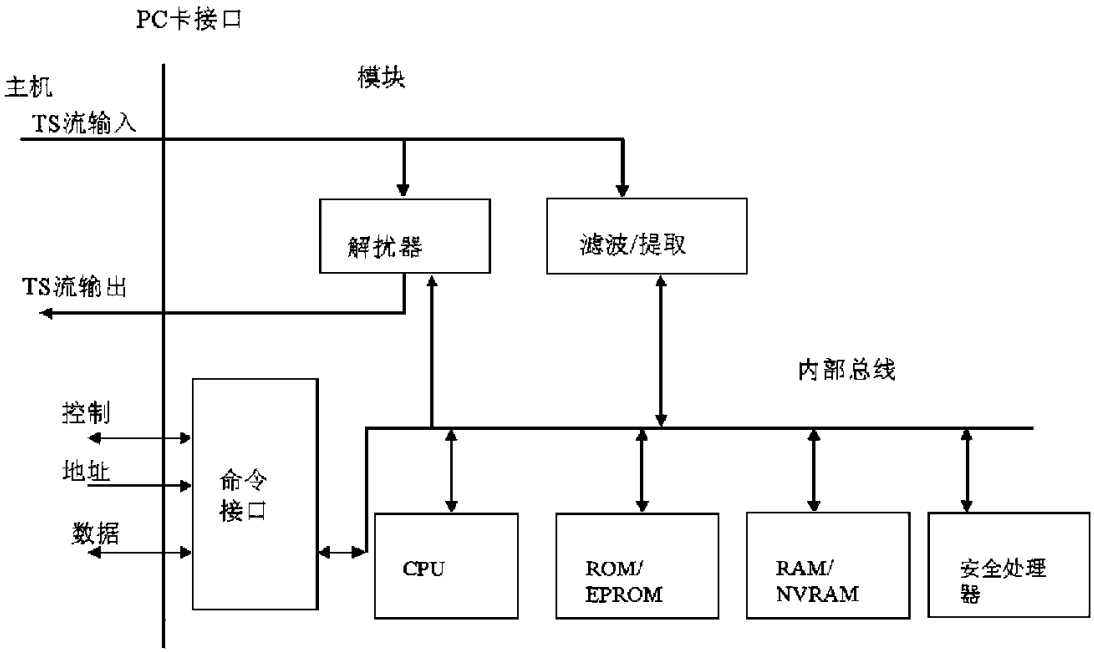


图 F18　典型的模块结构以及 PC 卡接口的位置

F7.1.1　PC 卡接口

在下面几个章节将描述，它是 PC 卡规范的变形。这个变形提供以下功能：

—MPEG-2 数据的 TS 流接口，包括一个 8 比特并行输入、单独的 8 比特并行输出、控制信号、字节时钟；

—传送命令的主机和模块之间的命令接口，包括 8 比特双向数据总线、控制信号、地址信号；

—允许主机读取卡的结构信息以及配置模块到正常工作模式的内存接口。

F7.1.2　解扰器

解扰器有选择性地对 TS 流的 TS 包进行解扰。如果配置成 PES 级别解扰，它也可以这样做。它由通过 CPU 周期性输入的控制字 CW 来配置。由于解扰器有可能需要同时解扰几个业务，因此其中预留有一个控制字 CW 与需要解扰的 PID 的对应列表。当 transport _ scrambling _ control 标记设置为 00 时，即便 PID 和当前的控制字可以匹配，解扰器不会去试图对任何 TS 包进行解扰。当使用 PES 级别解扰时，对 PES _ scrambling _ control 标记也有同样限制。

F7.1.3　滤波器/提取

CA 系统正常操作所需要的部分数据包含在 TS 流中。滤波器/提取电路配置为提取 CA 模块用于解扰相应节目/业务所需要的数据。

F7.1.4　CPU

负责运行 CA 进程，组织模块内以及模块和主机间的数据流程，以完成 CA 的功能。

F7.1.5　ROM/EPROM 和 RAM/NVRAM

其中包含完成 CA 处理进程所需要的程序、数据。在 ROM 中也可以包含 PC 卡初始化过程中所需要的特性参数。

F7.1.6　安全处理器

这是个独立的处理器，比主 CPU 具有更高的安全性要求，可以执行安全功能，例如解密、存储安全信息如密钥、授权等。它可以嵌入在 PC 卡模块内部，也可以驻留在附加的可分离的模块如智能卡中。

F7.2 电接口

这个接口的电特性满足 PC 卡标准 2.1 版本中的规范要求—见 F7.5。

F7.2.1 TS 流接口

主机提供给模块的 MPEG-2 数据接口为 8 比特的数据总线 MDI0—MDI7。另外还有两个控制信号 MISTRT 和 MIVAL。这些是通过 MCLK 时钟信号驱动的。模块中返回的 MPEG-2 数据从 8 比特数据总线 MD00—MD07 输出。类似地有两个控制信号 MOSTRT 和 MOVAL。对这个接口的详细描述可见 F7.5。

F7.2.2 命令接口

F7.2.2.1 硬件接口描述

硬件接口包括几个占用 PC 卡接口 4 字节地址的寄存器。偏移量为 0 的是数据寄存器，用来从模块读取或写入数据。偏移量为 1 的为控制寄存器和状态寄存器。在偏移量为 1 时，读取的是状态寄存器，写入的是控制寄存器。大小寄存器是偏移量为 2、3 的 16 比特寄存器，包括偏移量为 2 的低半部分 LS、偏移量为 3 的高半部分 MS。图 F19 给出了寄存器的地址对应图。接口只解码两个地址线 A0、A1。主机的设计者可以通过合适的其他数据线的解码、影射，把这四个字节的块放在它的自己的地址空间的合适地方。

偏移量	寄存器
0	数据寄存器
1	命令/状态寄存器
2	大小寄存器（LS）
3	大小寄存器（MS）

图 F19 硬件寄存器接口地址表

状态寄存器如图 F20 所示。

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

图 F20 状态寄存器

当模块有数据要发送给主机时设置 DA（数据可用）为 1。当模块空闲可以接收来自主机的数据时，或者在由模块硬件复位或 RS 命令引起的复位周期结束时，设置 FR（空闲）为 1。R 表示是预留比特，通常为 0。WE（写错误）和 RE（读错误）用来表示读写操作时的长度错误。

命令寄存器如图 F21 所示。

比特	7	6	5	4	3	2	1	0
	R	R	R	R	RS	SR	SW	HC

图 F21 命令寄存器

RS（复位）设为1表示需要对接口复位，它并不对整个模块进行复位。SR（读取大小）设为1时，请求模块提供它的最大缓冲区尺寸，当主机传送结束后将复位为0。SW（写入大小）设为1时，告诉模块使用哪个缓冲区大小，在数据传输结束后由主机复位到0。HC（主机控制）由主机在启动数据写入序列前设为1。当数据传输结束后，主机将把它复位到0。R表示为预留比特，它们通常为0。

F7.2.2.1.1 安装

在模块初始化过程中以及其他时候，当发生错误时，主机需要能够对接口进行复位。它通过向控制寄存器的RS比特写入1来实现。模块将清除数据传输缓冲区中的所有数据、设置接口，以便可以进行缓冲区大小的协商过程。当复位操作结束时，将设置FR比特为1以做出指示。

在初始化结束后，主机通过缓冲区大小协商协议来确定模块的内部缓冲区大小。在此协议结束前，主机、模块都不能使用这个接口。主机通过写入1到控制寄存器的SR比特来开始协商过程，然后等待DA比特被设为1，并从下面描述的模块传输给主机的数据中读取缓冲区大小。在传输操作结束后，主机将SR比特复位为0。返回的数据为2个字节。

模块至少应支持16字节的最小缓冲区大小。最大值由大小寄存器确定（65535字节）。类似地，主机可能也有一个缓冲区大小的限制。主机应最小支持256字节缓冲区大小，最大支持65535字节。在读取模块支持的缓冲区大小后，主机将取模块缓冲区大小和自己的缓冲区大小的较小的一个。这将是后续的主机和模块之间数据传输所用的缓冲区大小。主机通过向控制寄存器的SW比特写入1来通知模块使用该缓冲区大小，然后等待FR比特被设为1，然后再使用下面描述的主机和模块之间的传输方式把缓冲区大小以2字节数据的方式写入，其中权重最大的比特放在第一位。当传输结束时，主机把SW比特设为0。协商确定的缓冲区大小对双向数据传输都使用，甚至在双缓冲区实现情况下也是如此。

F7.2.2.1.2 主机向模块的传输

主机设置HC比特为1，然后测试FR比特。当FR是0时表示接口正忙，主机必须重新设置HC并等待一段时间再测试。如果FR为1，主机将向大小寄存器中写入想传送的数据的字节数，然后向数据寄存器中写入数据字节。这个多重写入过程不应被接口上的其他操作所打断，除了读取状态寄存器的操作之外。当写入第一个字节时，模块设置WE为1、FR为0。在传输过程中，WE保持为1。直至最后一个字节被写入，WE被设为0。如果还有字节需要写入，WE被设为1。在传输结束时，HC比特被设为0。主机必须在开始上述的主机—模块周期之前测试DA比特，以防止由于模块的单个缓冲区实现引起的死锁现象。

下面给出了主机侧的C语言代码片段：

```
if (Status _ Reg &. 0x80) / * 跳转到模块向主机传送 （见下） * /
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;
```

F7.2.2.1.3 模块向主机的传输

主机周期性地测试状态寄存器中的DA比特。当DA为1时，主机读取大小寄存器以确定传输数据的字节数。然后主机将从数据寄存器中读取数据比特。这个多重读取过程不应被接口上的其他操作打断，除了读取状态寄存器的操作之外。当读取第一个字节后，模块设置RE为1、DA为0。在传输过程最后一个字节被读取前RE比特保持为1，在读取结束后RE设为0。如果还有字节要读取，则RE设为1。下面的C语言编码给出了主机端的处理进程的例子：

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

大小寄存器的字节可以按任何顺序读写。注意这个接口不支持模块对主机的打断。主机将周期性地测试状态寄存器中的DA、FR比特，以决定是否需要通信。在本附录的将来的版本中，可能会增加对操作打断的支持。

F7.2.2.2 模块中的硬件支持

任何符合上述主机—模块互操作的规范的实现都是可能的。模块可以使用一个缓冲区来支持双向会话，也可以在每个方向上的数据传输使用一个缓冲区以加速接口速度。通过对控制缓冲区操作的HC、FR、DA比特的适当控制，两种接口设计方式都可以使用。

F7.3 链路层

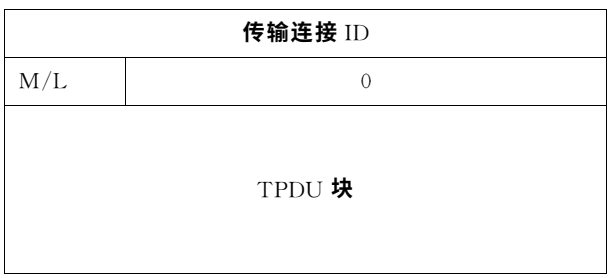
F7.3.1 TS流接口

在TS流接口上没有链路层。数据以连续的MPEG-2 TS包形式传送，在TS包内部、之间可能会有间隙。

F7.3.2 命令接口

命令接口的链路层完成两项工作。为了在物理层有限的缓冲区上传输传输协议数据单元TPDU，必要时它将TPDU分成块，并在接收端重新组合。它同时可以把几个传输连接复接到一个链路连接上去。它通过把所有想在链路上传输的块交织起来的方式来实现复接。物理层传输机制是可靠的，即保持数据的正确顺序、既不删除又不重复。

链路层在物理连接即插入模块或上电、读取卡结构、配置模块工作模式之后，将自动建立，不需要其他外在的过程。链路协议数据单元LPDU的大小取决于主机和模块在接口上使用SR、SW命令协商的大小。每个LPDU由2个字节的头、TPDU的块组成，总的大小不超出协商的缓冲区大小。头的第一个字节是TPDU块的传输连接标志。第二个字节的第一个比特包括更多/最后标志，如果为1表示其后至少还有一个TPDU块，如果为0表是这是该传输连接的最后（或仅有）的TPDU块。第二个字节的其他比特全为预留比特，应设为0。图F22说明了这种情况：



图F22 链路协议数据单元的布局

每个 TPDU 以一个新的 LPDU 开始，即 LPDU 若载有传输连接上的先前的 TPDU 的最后块，则不能同时载有下一个 TPDU 的第一个块。如果有多于一个传输连接当前有 TPDU 要发送，链路层将轮流发送每个 TPDU 的块，这样所有的传输连接将公平地分摊得到可用的带宽。

F7.4 PC 卡接口的传输子层的实现细节

F7.4.1 传输协议对象

命令接口上的传输协议为命令—响应协议，当主机以命令传输协议数据单元 C _ TPDU 发送命令给模块时，主机将等待模块使用响应传输协议数据单元 R _ TPDU 做出的应答。模块不能初始化通信过程：它必须等待主机查询或者主机先发送数据。这个协议由 11 个传输层对象支持。有些只在主机的 C _ TPDU 中出现，有的只在模块的 R _ TPDU 中出现，有的在两者中都出现。

Create _ T _ C 和 C _ T _ C _ Reply 建立新的传输连接；Delete _ T _ C 和 D _ T _ C _ Reply 清除连接；Request _ T _ C 和 New _ T _ C 允许模块请求主机建立一个新的传输连接；T _ C _ Error 用来指示错误条件；T _ SB 载有模块提供给主机的状态信息；T _ RCV 接收来自模块的数据，T _ Data _ More 和 T _ Data _ Last 在模块、主机之间的较高层次上传输数据。当主机没有数据发送时，它可以使用空的数据域的 T _ Data _ Last 来对模块进行查询。在所有的对象中，都有编码的 tag _ field、length _ field 域，以及一个单独的 t _ c _ id 字节。length _ field 域的编码见表 F1。

F7.4.1.1 命令 TPDU

命令 TPDU（C _ TPDU）载有从主机到模块的传输协议对象。见图 F23。

头	主体内容
c _ tpdu _ tag length _ field t _ c _ id	[数据域]

图 F23 C _ TPDU 的结构

C _ TPDU 编码见表 F80。

表 F80 C _ TPDU 编码

句法	比特数	类型
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
}		
}		

C _ TPDU 由两部分组成：

一个头，包括 TPDU 编码的标志 c _ tpdu _ tag、表示后面域的长度的 length _ field、以及称为传输连接标志的 t _ c _ id。

一个可变长度的主体内容域，其长度为 length _ field－1。

F7.4.1.2 响应 TPDU

响应 TPDU （R _ TPDU）载有模块向主机传送的传输协议对象。如图 F24。

头	主体内容	状态
R _ tpdu _ tag length _ field t _ c _ id	[数据域]	SB _ tag length _ field＝2 t _ c _ id SB _ value

图 F24 R _ TPDU 结构

R _ TPDU 编码见 F81。

表 F81 R _ TPDU 编码

句法	比特数	类型
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		
t _ c _ id	8	uimsbf
SB _ value	8	uimsbf
}		

R _ TPDU 由三部分组成：

一个头，包括 TPDU 编码的标志 r _ tpdu _ tag、表示后面传输连接标志和数据域长度的 length _ field、称为 t _ c _ id 的传输连接标志。在长度域 length _ field 中未计算状态的字节数；

一个长度等于 length _ field－1 的可变长度的主体内容；

一个由状态标记 SB _ tag、等于 2 的 length _ field、一个传输连接标志、一个 1 字节的按照图 F25 组成的必须状态域。

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
DA	预留						

图 F25 SB _ value 编码

1 比特的 DA（数据可用）标志域指示模块在输出缓冲区中是否有要发送给主机的消息。主机必须发送 Receive _ data 接收数据 C _ TPDU 来接收消息（见 F7.4.1.11.2）。DA 域的编码在表 F82 中给出。

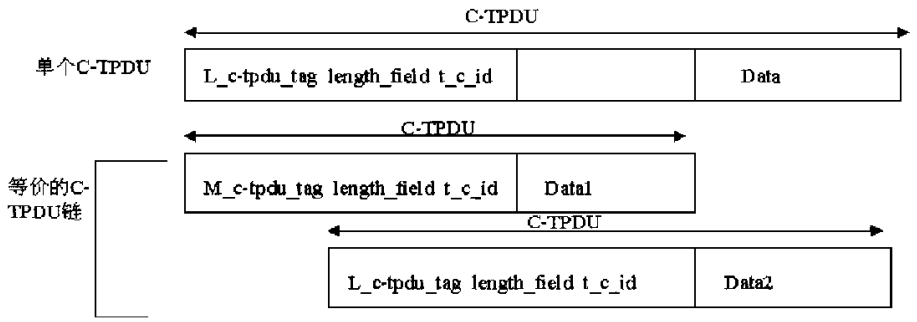
“reserved”预留值应设为0。

表F82 SB_value的比特8的编码

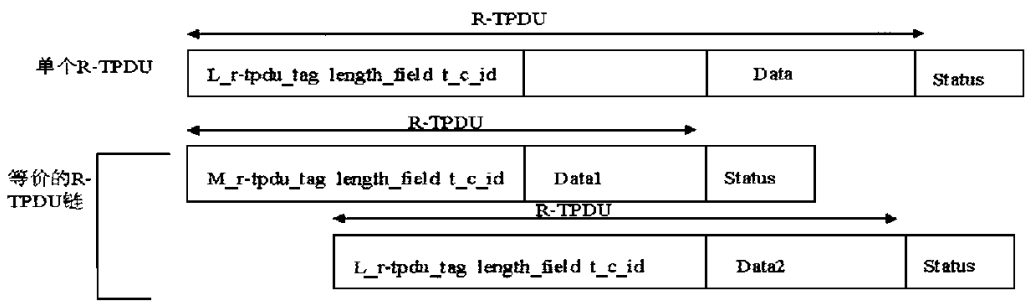
比特8	含义
0	没有消息
1	有消息

F7.4.1.3 命令或响应TPDU数据域的链接

如果模块或主机的发送和接收缓冲区大小要求，在C_TPDU或R_TPDU中的数据域应分割成较小的块。这个改变是通过在C_TPDU中使用两个不同的c_TPDU_tag值（M_c_TPDU_tag、L_c_TPDU_tag）、在R_TPDU中使用两个不同的r_TPDU_tag值（M_r_TPDU_tag、L_r_TPDU_tag）来实现的。数据域的所有块，除了最后一个，都使用M_c_TPDU_tag（或M_r_TPDU_tag）值的C_TPDU（或R_TPDU）发送。这个标记值表示将在另一个C_TPDU（或R_TPDU）中发送更多的数据。数据域的最后块使用L_c_TPDU_tag（或L_r_TPDU_tag）值的C_TPDU（或R_TPDU）发送。当接收到最后一个块时，接收方把所有接收的数据合并起来。这个机制对所有的具有两个标记值的C_TPDU（或R_TPDU）都是有效的。下面的图F26和图F27对此做出了说明。



图F26 C_TPDU链机制示意图



图F27 R_TPDU链机制示意图

F7.4.1.4 建立传输连接Create_T_C

Create _ T _ C 的结构如图 F28。

头

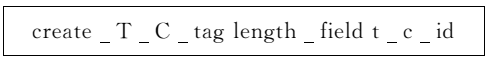


图 F28 Create _ T _ C 的结构

Create _ T _ C 编码见表 F83。

表 F83 Create _ T _ C 编码

句法	比特数	类型
Create _ T _ C () {		
create _ T _ C _ tag	8	uimsbf
length _ field () =1		
t _ c _ id	8	uimsbf
}		

Create _ T _ C 对象只有一个部分组成：

一个头，包括一个 Create _ T _ C 对象编码的标记值、一个等于 1 的 length _ field、一个称为 t _ c _ id 的传输连接标志。

F7.4.1.5 建立传输连接应答 C _ T _ C _ Reply

C _ T _ C _ Reply 的结构如图 F29 所示。

头

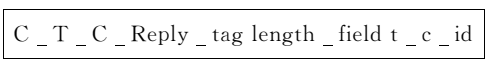


图 F29 C _ T _ C _ Reply 的结构

C _ T _ C _ Reply 编码见表 F84。

表 F84 C _ T _ C _ Reply 编码

句法	比特数	类型
C _ T _ C _ Reply () {		
C _ T _ C _ Reply _ tag	8	uimsbf
length _ field () =1		
t _ c _ id	8	uimsbf
}		

C _ T _ C _ Reply 对象只由一部分组成：

一个头，包括一个对 C _ T _ C _ Reply 编码的 C _ T _ C _ Reply _ tag 标记值、一个等于 1 的 length _ field、一个传输连接标志。

F7.4.1.6 删除传输连接 Delete _ T _ C

Delete _ T _ C 的结构如图 F30 所示。

头

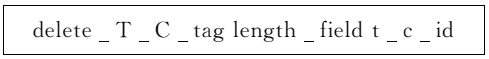


图 F30 Delete _ T _ C 的结构

Delete _ T _ C 编码见表 F85。

表 F85 Delete _ T _ C 编码

句法	比特数	类型
Delete _ T _ C () {		
delete _ T _ C _ tag	8	uimsbf
length _ field () =1		
t _ c _ id	8	uimsbf
}		

Delete _ T _ C 对象只有一个部分组成：

一个头，包括 Delete _ T _ C 对象编码的标记值、一个等于 1 的 length _ field、一个传输连接标志。

F7.4.1.7 删除传输连接应答 D _ T _ C _ Reply

D _ T _ C _ Reply 的结构如图 F31 所示。

头

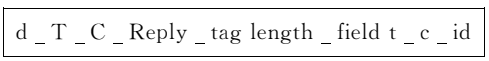


图 F31 D _ T _ C _ Reply 的结构

D _ T _ C _ Reply 编码见表 F86。

表 F86 D _ T _ C _ Reply 编码

句法	比特数	类型
D _ T _ C _ Reply () {		
d _ T _ C _ Reply _ tag	8	uimsbf
length _ field () =1		
t _ c _ id	8	uimsbf
}		

D _ T _ C _ Reply 对象只包含一个部分：

一个头，包括对 D _ T _ C _ Reply 对象编码的标记值、一个等于 1 的 length _ field、一个传输连接标志。

F7.4.1.8 请求传输连接 Request _ T _ C

Request _ T _ C 的结构如图 F32。

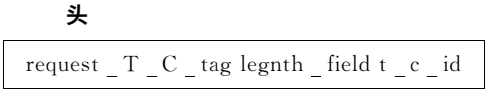


图 F32 Request _ T _ C 的结构

Request _ T _ C 编码见表 F87。

表 F87 Request _ T _ C 编码

句法	比特数	类型
Request _ T _ C () {		
request _ T _ C _ tag	8	uimsbf
length _ field () =1		
t _ c _ id	8	uimsbf
}		

Request _ T _ C 对象只包含一部分：

一个头，包括对 Request _ T _ C 对象编码的 request _ T _ C _ tag 标记值、一个等于 1 的 length _ field、一个传输连接标志。

F7.4.1.9 新传输连接 New _ T _ C

New _ T _ C 的结构如图 F33 所示。

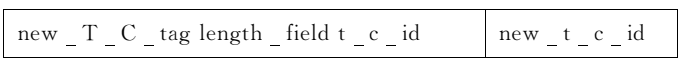


图 F33 New _ T _ C 的结构

New _ T _ C 编码见表 F88。

表 F88 New _ T _ C 编码

句法	比特数	类型
New _ T _ C () {		
new _ T _ C _ tag	8	uimsbf
length _ field () = 2		
t _ c _ id	8	uimsbf
new _ t _ c _ id	8	uimsbf
}		

New _ T _ C 对象由两部分组成：

一个头，包括对 New _ T _ C 对象编码的 new _ T _ C _ tag 标记值、一个等于 2 的 length _ field、一个传输连接标志；

一个包含将要建立的新连接的传输连接标志的必须主题标志。

F7.4.1.10 传输连接错误 T_C_Error

T_C_Error 的结构如图 F34 所示。

头	主体内容
T_C_Error_tag length_field t_c_id	error_code

图 F34 T_C_Error 的结构

T_C_Error 编码见表 F89。

表 F89 T_C_Error 编码

句法	比特数	类型
T_C_Error () {		
T_C_Error_tag	8	uimsbf
length_field () = 2		
t_c_id	8	uimsbf
error_code	8	uimsbf
}		

T_C_Error 对象由两部分组成：

一个头，包括对 T_C_Error 对象编码的 T_C_Error_tag 的标记值、一个等于 2 的 length_field、一个传输连接标志；

一个必须的主题内容，包含了所发生的特定错误的错误代码。错误码编码见表 F90。

表 F90 错误编码值

错误编码	含义
1	没有可用的传输连接

F7.4.1.11 C_TPDU 和相关 R_TPDU 的列表

F7.4.1.11.1 发送数据命令

这个命令由主机使用。在传输连接处于打开状态下时，主机用来发送数据给模块或从状态字节中获得状态信息。模块以状态字节应答。如图 F35 所示。

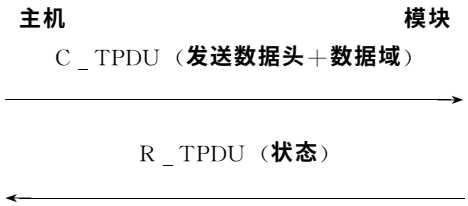


图 F35 发送数据命令/响应对

发送数据 C _ TPDU 的编码见表 F91。

表 F91 发送数据 C _ TPDU 的编码

c _ TPDU _ tag	M _ c _ TPDU _ tag：T data _ more L _ c _ TPDU _ tag：T data _ last
length _ field 长度域	数据域的长度［3］
t _ c _ id	传输连接标志
data field 数据域	（TLV TLV…. TLV）的子集

发送数据 R _ TPDU 的编码见表 F92。

表 F92 发送数据 R _ TPDU 的编码

状态	见图 F25

L＝1（没有数据域）的发送数据 C _ TPDU 可以由主机用来从状态字节获得状态信息。

F7.4.1.11.2 接收数据命令

主机使用这个命令从模块接收数据。如图 F36 所示。

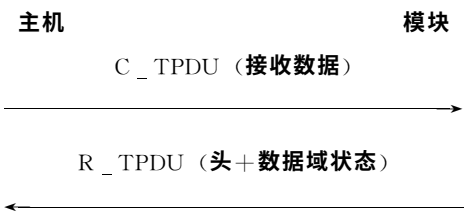


图 F36 接收数据命令/响应对

接收数据 C _ TPDU 编码见表 F93。

表 F93 接收数据 C _ TPDU 编码

c _ TPDU _ tag	T RCV
length _ field	c _ TPDU _ length 设为 1
T _ c _ id	传输连接标志

接收数据 R _ TPDU 编码见表 F94。

表 F94 接收数据 R _ TPDU 编码

r _ TPDU _ tag	M _ r _ TPDU _ tag：T data _ more L _ r _ TPDU _ tag：T data _ last
length _ field	数据域的长度
T _ c _ id	传输连接标志
数据域	（TLV TLV…TLV）子集
状态	见图 F25

F7.4.1.12 查询功能的规则

查询功能向模块发送命令，以检查它是否有数据要向主机发送。主机通常用这个功能来发送长度L=1的C_TPDU（只有t_c_id域，没有数据域）。如果数据不可用，模块以DA为0的状态字节来应答。如果数据可用，模块以DA为1的状态字节来应答。主机然后可以发送1或多个接收数据C_TPDU，来激发数据传输过程，直至状态字节中的DA被重新设置为0。当模块还有数据要发送，主机接收到DA为1的状态字节，则查询功能将被暂停。查询功能在接收到DA为0的状态字节后将重新开始。查询功能的最大周期的100 ms。对每个查询功能，都设有300 ms的超时界限，当时间耗尽时，主机将删除传输连接并返回到正常状态。在等待查询应答时，主机不再发送任何附加的查询，甚至在超出通常的查询间隙时也是如此。

F7.4.1.13 传输标记列表

Tpdu_tag的编码遵循ASN.1中的规则。每个tpdu_tag采用1字节编码。见表F95。

表F95 传输标记的编码

tpdu_tag	标记值（HEX）	原始或构造的（P或C）	方向（主机<——>模块）
T SB	80	P	<————
T RCV	81	P	————>
T create_t_c	82	P	————>
T c_t_c_reply	83	P	<————
T delete_t_c	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	<————>

F7.5 供符合规范的主机和模块使用的PC卡子集

本部分定义了主机、模块使用的PC卡规范的最小子集。本部分并不限制主机超出最小子集的内容，例如支持其他类型的PC卡设备。然而，所有的模块必须设计为使符合最小子集规范主机可以和它正常操作。本部分规范对PC卡规范做出了相应扩展 。

在本部分使用了十六进制的数字，它的格式是最后六个数字使用大写的A—F，再加上小写的h，例如7FFh。

F7.5.1 物理规范

F7.5.1.1 卡的外形尺寸

主机应支持PC卡标准，第3卷中描述的类型I和类型II的PC卡。其中，主机对类型III的卡的支持是可选要求。

F7.5.1.2 连接器

参见PC卡标准，第3卷中4节。

F7.5.1.3 PC 卡指导

参见 PC 卡标准，第 3 卷中 5 节。

F7.5.1.4 接地/EMI

参见 PC 卡标准，第 3 卷中 6 节。

F7.5.1.5 连接器的可靠性

参见 PC 卡标准，第 3 卷中 7 节。

F7.5.1.6 连接器的持久性

参见 PC 卡标准，第 3 卷中的 8.2 节（苛刻环境下）。

F7.5.1.7 PC 卡环境

参见 PC 卡标准，第 3 卷中的 9 节。关于温度的规范要求，PC 卡应能在 55 C 下工作。模块应能支持可靠的电源或电池操作。为了使得这个操作环境的限制更容易实现，主机应在模块在额定功率下工作时，把主机周围环境的温度与模块周围的温度之差限制在 15 C 内。注意这并不能保证模块在所有的环境下保持在主机可接受的 55 C 以下。模块设计者必须遵循关于电池放置的 PC 卡规范以使得含有电池的模块的误操作的可能最小化，主机设计者在主机热性能方面设计时应考虑到这些建议。需要注意的是，设计者必须遵循对模块温度的任何强制性安全规范的建议。

F7.5.2 电规范

CI 接口的模块是 16 比特 PC 卡电接口（参见 PC 卡标准，第 2 卷中 4 节）实现的变形。命令接口使用数据总线的低字节、地址总线的 A0—A14、以及合适的控制信号。命令接口以 I/O 接口模式方式工作。高地址线 A15—A25、数据总线的 D8—D15、以及其他控制信号对此接口变形进行重新定义。

当第一次插入，在被配置之前，除了以下限制之外，遵循本接口规范的模块就象一个存储器设备一样：

a）D8—D15 保持在高阻状态；

b）16 比特读写操作模式不可用。CE2＃将被模块忽略，并解释成处于“高”电平；

c）地址线 A15—A25 不能用做地址信号。模块上最大可寻址的地址空间为 32 768 字节（16 384 字节的只出现在偶地址的存储器空间）；

d）BVD1、BVD2 信号保持高电平。

F7.5.2.1 卡类型检测

参见 PC 卡标准，第 2 卷中的 3 节。主机应支持 5 v 电压工作，作为可选项应支持 3.3 v 电压工作。在后者情况下，主机将使用 PC 卡标准，第 2 卷中的检测机制来决定操作电压。根据是否提供在 3.3 v 电压下工作的能力，主机将遵循 PC 卡标准，第 2 卷中 Section 3.2 的插槽规范。主机不一定需要支持 PC 卡标准，第 2 卷中 3.3 节中的 PC 卡总线检测机制，但是作为可选项它可以遵循。

F7.5.2.2 插针分配

当在复位后并处于存储器卡模式时，适用于 PC 卡标准，第 2 卷中表 4.1、4.2 左边一列的插针分配。当在初始化过程中模块作为 CI 接口变形时，采用以下的插针重分配方案：载有 A15—A25、D8—D15、BVD1、BVD2、VS2＃信号的插针用来提供 MPEG-2 复接数据的高速输入和输出总线。除了没有 IOIS16＃、CE2＃被忽视之外，所有其他的插针和作为 I/O 和存储器接口时的分配方案相同。

下面的表 F96 给出了通常的用户接口的插针分配方案。

表 F96 PC 卡的插针分配

插针	I/O	信号	功能	插针	I/O	信号	功能
1	GND		Ground	35	GND		Ground
2	D3	I/O	数据位 3	36	CD1#	O	卡检测 1
3	D4	I/O	数据位 4	37	MDO3	O	MP 数据输出 3
4	D5	I/O	数据位 5	38	MDO4	O	MP 数据输出 4
5	D6	I/O	数据位 6	39	MDO5	O	MP 数据输出 5
6	D7	I/O	数据位 7	40	MDO6	O	MP 数据输出 6
7	CE1#	I	卡使能 1	41	MDO7	O	MP 数据输出 7
8	A10	I	地址位 10	42	CE2#	I	卡使能 2
9	0E#	I	输出使能	43	VS1#	O	电压感应 1
10	A11	I	地址位 11	44	IORD#	I	I/O 读
11	A9	I	地址位 9	45	IOWR#	I	I/O 写
12	A8	I	地址位 8	46	MISTRT	I	MP 正在启动
13	A13	I	地址位 13	47	MDI0	I	MP 数据输入 0
14	A14	I	地址位 14	48	MDI1	I	MP 数据输入 1
15	WE#	I	写使能	49	MDI2	I	MP 数据输入 2
16	IREQ#	O	中断请求	50	MDI3	I	MP 数据输入 3
17	VCC		Vcc	51	VCC		Vcc
18	VPP1		Program voltage 1	52	VPP2		Program voltage 2
19	MIVAL	I	有效 MP 输入	53	MDI4	I	MP 数据输入 4
20	MCLKI	I	MPEG-2 时钟输入	54	MDI5	I	MP 数据输入 5
21	A12	I	地址位 12	55	MDI6	I	MP 数据输入 6
22	A7	I	地址位 7	56	MDI7	I	MP 数据输入 7
23	A6	I	地址位 6	57	MCLKO	O	MPEG-2 时钟输出
24	A5	I	地址位 5	58	RESET	I	卡复位
25	A4	I	地址位 4	59	WAIT#	O	扩展总线周期
26	A3	I	地址位 3	60	INPACK#	O	输入口确认
27	A2	I	地址位 2	61	REG#	I	寄存器选择
28	A1	I	地址位 1	62	MOVAL	O	MP 输出有效
29	A0	I	地址位 0	63	MOSTRT	O	MP 输出开始
30	D0	I/O	数据位 0	64	MDO0	O	MP 数据输出 0
31	D1	I/O	数据位 1	65	MDO1	O	MP 数据输出 1
32	D2	I/O	数据位 2	66	MDO2	O	MP 数据输出 2

表 F96 PC 卡的插针分配（完）

插针	I/O	信号	功能		插针	I/O	信号	功能
33	IOIS16#		16 比特 I/O（总是高电平）		67	CD2#	O	卡检测 2
34	GND		地		68	GND		地

F7.5.2.3 16—比特 PC 卡特性

参见 PC 卡标准，第 2 卷中的 4.3 节。在内存存取过程中，模块至少应支持 12 比特的地址解码（4 096字节）。在 I/O 过程中，可以解码较少比特。

F7.5.2.4 信号描述

参见 PC 卡标准，第 2 卷中 4.4、4.6、4.7 节。下面的信息对本附录做出补充：

提供 MPEG-2 TS 流接口的输入、输出总线（MDI0—7，MDO0—7），同时提供 MCLKI、MCLKO、MISTRT、MIVAL、MOSTRT、MOVAL 控制信号。

MCLKI 为提供给模块的 MDI0—7 总线上的字节提供时钟，MCLKO 为模块输出总线 MDO0—7 提供时钟周期。对传送 TS 流的大多数模块，MCLKO 通常只是 MCLKI 经过一段延迟的版本。对发起数据的模块例如单独的前端或网络连接，MCLKO 可以从数据中提取。图 F37 给出了 MPEG-2 TS 流接口、MCLKI、MCLKO 以及数据信号之间的关系，而表 F97 对这些时序关系做出了限制。注意输出时序的限制通常可以很容易地从 MCLKO 的下降沿产生输出来满足。对 MCLKI、MCLKO 之间的延迟，没有进一步规范。当模块提供自己的 MCLKO 时，它们甚至可以采用不同频率。主机将以雏菊链（Daisy-Chain）的形式将一个模块的 MCLKO 连接到另一个模块的 MCLKI，以支持多个 CI 接口插槽。主机其余部分的 TS 流参考时钟从最后一个接口插槽的 MCLKO 中提取。MCLKI 和 MCLKO 都应该是连续的。采用这种脉冲时钟方式并不是有意的。脉冲数据通过对 MIVAL 和 MOVAL 的合适使用来控制。

在 MDI0—7 上传输每个传输包的第一个字节时，MISTRT 总是有效的。这个信号的上升、下降沿时序关系和 MDI0—7 相同。

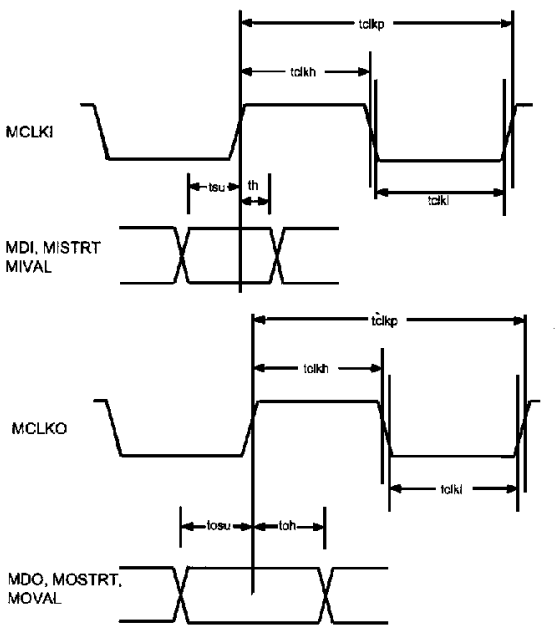


图 F37 TS 流接口的时序关系

表 F97 时序关系限制

项目	符号	最小值	最大值
时钟周期	Tclkp	111 ns	
时钟高电平期	Tclkh	40 ns	
时钟低电平期	Tclkl	40 ns	
输入数据建立期	Tsu	15 ns	
输入数据保持期	Th	10 ns	
输出数据建立期	Tosu	20 ns	
输出数据保持期	Toh	15 ns	

MIVAL 表示 MDI0－7 上的数据有效。TS 包上的所有字节时间上可以是连续的，在此期间 MIVAL 都为逻辑 1。不管主机采用何种时钟策略，在 TS 包内部或 TS 包之间的一些连续字节之间会存在一个或多个字节间隙。MIVAL 在这一个或多个字节间隙期间返回到 0，表示这些数据可以被忽略。

MDO0－7 是 MPEG-2 TS 流数据的输出总线。这里 MCLKO 从 MCLKI 得到，对应的 MDO0－7 上的数据是 MDI0－7 上延迟的和可能已解扰的数据。

MOSTRT 在 TS 包输出的第一个字节期间有效。

MOVAL 以和 MIVAL 类似方式，指示 MDO0－7 上的字节为有效。MOVAL 不必是 MIVAL 延时。PC 卡标准，第 2 卷中的 4.4.7 节中定义的对主机中断请求的支持是可选项。这个功能不应被 CA 模块所使用。模块对中断的支持在将来版本中加入，主机设计者可以注意这个版本的发布。无论何时当模块对 I/O 操作做出响应时，将都称为是 INPACK＃（参见 PC 卡标准，第 2 卷中的 4.4.22 节）。

F7.5.2.5 内存功能

参见 PC 卡标准，第 2 卷中的 4.6 节。主机对内存功能的支持是必须的。注意内存存储是以字节为单位的，其宽度为字节宽度，数据只出现在 D7－D0 上。连续的字节将出现在连续的偶地址上（0、2、4…等）。当模块配置为和 CI 接口进行操作时，内存应仍然可以进行读写操作。主机对公共内存（Common Memory）的功能的支持是可选的，而模块不应使用此公共内存。

F7.5.2.6 时序功能

参见 PC 卡标准，第 2 卷中的 4.7 节。主机必须支持属性内存（Attribute Memory），而对公共内存的支持是可选的。

F7.5.2.7 电接口

参见 PC 卡标准，第 2 卷中的 4.9 节。4.9.3.2 节中定义的对重叠 I/O 地址窗口，主机对其是否支持是可选的。模块应使用大小为 4 字节的独立的 I/O 地址窗口。

F7.5.2.8 卡检测

参见 PC 卡标准，第 2 卷的 4.10 节。

F7.5.2.9 电池电压检测

模块不应实现或需要这种功能。主机对其的支持也是可选的。

F7.5.2.10 上电和关机

参见 PC 卡标准，第 2 卷中的 4.12 节，包括在配置时平均电流的规范。模块应明确在 CIS 中的操

作电流。每个模块所耗的功率不应超出 1.5 瓦。为每个模块额外供电电源，（Vcc 电流和 Vpp 电流之和）长期值不应超出 300 mA，短期峰值不应在 1 ms 内连续超出 500 mA。主机需要支持 Vpp 电压值，如果它们支持，则其电压和 Vcc 相同。

F7.5.2.11 I/O 功能

参见 PC 卡标准，第 2 卷中的 4.13 节，不同的是模块应使用 8 比特读写模式。

F7.5.2.12 功能配置

参见 PC 卡标准，第 2 卷中 4.14 节。模块应只支持配置选项寄存器。对配置选项寄存器之外的其他寄存器的支持是可选的。

F7.5.2.13 卡配置

参见 PC 卡标准，第 2 卷的 4.15、4.15.1 节。

F7.5.3 元格式（Metaformat）规范

本部分定义了 CI 接口模块在卡信息结构中必须具有的最小元组集合，这也是主机需要用于识别的最小集合。可参见 PC 卡标准，第 4 卷中的 1、2、3 节。下面的元组是 CI 接口所使用的最小集合，也是主机识别所需的集合。由元组构成的元组链必须位于属性内存的第一个，虽然其后可以有其他内容，但是后续内容主机将不需要识别。主机必须考虑到元组链后跟着其他元组的情况。元组链一般以 CISTPL_NOLINK 元组结尾。可以参见 PC 卡标准，第 4 卷中的描述。

a）CISTPL_DEVICE_OA：按照 3.2.3 和 3.2.4 编码；

b）CISTPL_DEVICE_OC：按照 3.2.3 和 3.2.4 编码；

c）CISTPL_VERS_1：TPLLV1_MAJOR＝05h；TPLLV1_MINOR＝00h；其他域由模块厂商模块 3.2.10 定义；

d）CISTPL_MANFID：由厂商根据 3.2.9 定义；

e）CISTPL_CONFIG：由厂商根据 3.3.4 定义。配置寄存器基地址不应比 FEEh 大。在 TPCC_SBTPL 域中应包含一个子元组（见 3.3.4.5）。在子元组中，STCI_IFN 包括这种类型的 PC 卡发出的 0241h 作为标志 ID，STCI_STR 应包含字符串 DVB_CI_V1.00。STCI_IFN 的值表明这是一个 CI 接口兼容的模块。STCI_STR 字符串表示版本号。字符串的最后 5 个字符应是 Vx.xx，这里 x 是个数字，表示模块遵守的规范版本号；

f）CISTPL_CFTABLE_ENTRY：根据 3.3.2 定义。模块应支持至少 1 个 CISTPL_CFTABLE_ENTRY。对第一个入口，TPCE_INDX 的比特 6（缺省）、比特 7（接口）设为 1。配置的入口数可以是除 0 之外的任何值。TPCE_IF＝04h 表示用户接口 0。TPCE_FS 表示存在 I/O、电源配置入口。TPCE_IO 是一个等于 22h 的 1 字节的域—参见 PC 卡标准，第 4 卷 3.3.2.6。这个信息表示：模块使用 8 比特通道、对 2 条地址线解码。本节中“上电和关电”的电源配置入口，应遵循 PC 卡规范。Cftable 入口也包含下面两个子元组：

1）STCE_EV：参见 PC 卡标准，第 4 卷 3.3.2.10.1 节；

2）STCE_PD：参见 PC 卡标准，第 4 卷 3.3.2.10.2 节；

g）CISTPL_END：值为 FFh。如果 CA 模块包括除了上述定义的元组以外的部分，则在这些元组后应跟上 CISTPL_END。

F8 可选的附加对象

F8.1 确认

这个可选资源包含在主机中，以支持主机、单独的模块之间的确认过程，并使得确认过程在CA系统运营商的控制下完成。不想使用确认协议的广播者和/或CA系统运营商，必须能够在CI接口上传输信号，而不受到确认过程的干扰。

这个资源包括两个对象：用来实现确认过程的确认请求和确认响应。CA系统厂商和主机厂商之间的特别的使用方式在这里并未定义。

F8.1.1 确认请求和确认响应

这两个对象除了标记值以外，都是一样的。见表F98和表F99。

表F98 确认请求句法

句法	比特数	类型
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
}		
}		

表F99 确认响应句法

句法	比特数	类型
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
}		
}		

F8.1.2 确认资源编码

见表F100和表F101。

表F100 确认资源编码

资源	级别	类型	版本	资源ID
确认	16	1	1	00100041

表 F101 Apdu _ tag 编码

Apdu _ tag	标记值（HEX）	资源	方向（主机<——>应用）
T auth _ req	9F 82 00	确认	<———
T auth _ resp	9F 82 01	确认	———>

F8.2 电视图文显示资源

参照 GB/T 14219，中文图文电视广播规范。

F8.3 智能卡读取器资源级别

本部分描述了由主机或其他模块提供的可选的智能卡读取器资源。这个资源使用卡指令级别的命令（头、数据、状态字节）。命令在 GB/T 16649 中进行了定义。一共定义了四个对象。智能卡命令 Smart Card Cmd 和 Smart Card Reply 智能卡响应完成控制、响应功能。Smart Card Send 智能卡发送、Smart Card Rcv 智能卡接收完成数据的发送和接收功能。这个资源可以允许有多个会话建立连接，这样不同的应用可以对资源进行查询，但任何时候只有一个会话能处于“已连接”状态。这个资源是为了在主机或模块中为短期会话提供智能卡读取器，以完成银行卡、电视购物、PPV 付费收看等功能。这里并未做出对连续的安全性操作、加扰业务的收看等提供有关智能卡的长期会话的建议。这是因为智能卡读写器资源将被绑定并不能被其他应用所用，也不能对是否满足响应延迟的要求做出保证。

F8.3.1 对象

F8.3.1.1 智能卡命令

见表 F102 和表 F103。

表 F102 智能卡命令句法

句法	比特数	类型
smart _ card _ cmd () {		
smart _ card _ cmd _ tag	24	uimsbf
length _ field ()		
smart _ card _ cmd _ id	8	uimsbf
}		

表 F103 智能卡命令识别号

smart _ card _ cmd _ id	id 取值
Connect	01
Disconnect	02
power _ on _ card	03
power _ off _ card	04
reset _ card	05
read _ status	06
read _ answ _ to _ reset	07
Reserved	其他值

一旦和资源建立了会话，应用将发出这个命令。智能卡以资源返回的 Smart Card Reply 智能卡响应对象作为应答。下面列出了可用的命令：

connect：连接这个会话到卡读取器。如果连接成功，则响应 ID 为“已连接”，并返回适当的卡状态—“卡已插入、没有卡”等。如果卡读取器已经连接到另一个会话，返回的响应 ID 将为“忙”；

disconnect：取消卡读取器和本会话的连接。卡的电源将被关闭。通过适合的卡状态，返回的响应 ID 为“空闲”。如果卡读取器已经和另一个会话建立连接，返回的响应 ID 为“忙”；

power _ on _ card：通过接口设备打开并激活卡接口（Vcc 遵循 GB/T 16649 规范）。这个命令将对卡复位。如果卡已经在会话中建立连接，这个命令只允许执行一次。如果读取器未连接，将返回“空闲”；如果读取器已连接到另一个会话将返回“忙”；如果已经插入卡并进行了正确的复位，将返回“answ _ to _ reset”；如果没有卡插入或者卡未正确复位，将返回“no _ answ _ to _ reset”；

power _ off _ card：按照 GB/T 16649 规范关闭卡接口。如果卡已连接到一个会话并且此操作成功，将返回“已连接”；如果卡未连接到会话，将返回“空闲”；如果卡连接到另一个会话将返回“忙”；

reset _ card：当智能卡处于打开状态时对卡接口进行复位。其响应和 power _ on _ card 的响应相同；

read _ status：读取当前连接、卡的状态。发出此命令时，不一定必须建立连接；

read _ answ _ to _ reset：如果接口中有的话，读取当前卡的 answer-to-reset 消息。发出此命令时，不一定必须建立连接。当有 answer-to-reset 消息时，其响应为 answer-to-reset；否则返回 no _ answ _ to _ reset。

F8.3.1.2 智能卡响应

见表 F104、F105、F106。

表 F104 智能卡响应命令句法

句法	比特数	类型
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
}		
}		
}		

表 F105 智能卡响应识别码

smart _ card _ reply _ id	Id 取值
connected	01
free	02
busy	03
answ _ to _ reset	04
no _ answ _ to _ reset	05
reserved	其他值

表 F106 智能卡状态

smart _ card _ status	取值
card _ inserted	01
card _ removed	02
card _ in _ place _ power _ off	03
card _ in _ place _ power _ on	04
no _ card	05
unresponsive _ card	06
refused _ card	07
Reserved	其他值

Smart Card Reply 智能卡响应作为对 Smart Card Cmd 智能卡命令对象的应答，或者作为不合适操作的 Smart Card Send 智能卡发送对象的应答。在已连接的会话中，如果插入或去除卡，将也会发送此对象。响应 ID 取值如下：

connected：当不需要发送 answ _ to _ reset 或 no _ answ _ to _ reset 时，作为已连接会话中的响应；

free：作为对“拆除连接”命令的响应，或当智能卡读取器未连接时作为对其他命令的响应；

busy：当智能卡读取器已经和其他会话建立建立，发送此状态以表明不能对智能卡执行响应命令；

answ _ to _ reset：和智能卡中的 answer-to-reset 消息一起发送；

no _ answ _ to _ reset：当一个 answer-to-reset 消息被请求但不可用时发送。每个响应含有当前时刻的卡的状态。可以使用以下的卡状态值；

card _ inserted：当卡已插入时，未经请求就可以在会话上发送；

card _ removed：当卡被去除时，未经请求就可以在会话上发送；

card _ in _ place _ power _ off：当卡已插入并已被识别、但未上电的情况下，在所有的响应中发送；

card _ in _ place _ power _ on：当卡已插入并已被识别、已上电的情况下，在所有的响应中发送；

no _ card：当没有卡插入时在所有的响应中发送；

unresponsive _ card：当插入的卡不能对复位做出响应或者对 Smart Card Send 智能卡发送对象做出响应时，在所有的响应中发送；

refused _ card：当卡对复位做出响应但未按照预期方式进行的情况下，在所有的响应中发送。

F8.3.1.3 智能卡发送

见表 F107。

表 F107 智能卡发送句法

句法	比特数	类型
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
}		

这用来向已经结束了复位操作的处于连接状态的卡发送数据。其响应可以是含有数据的 Smart Card Rcv 接收对象和/或状态字节 SW1 和 SW2、或当其时不适合向智能卡发送数据时以 Smart Card Reply 返回状态信息。

F8.3.1.4 智能卡接收数据

见表 F108。

表 F108 智能卡接收句法

句法	比特数	类型
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
}		

这作为对处于连接状态的会话中的正在操作的卡的智能卡发送 Smart Card Send 的响应来发送。

F8.3.2 智能卡读取器资源编码

见表 F109 和 F110。

表 F109 智能卡读取器资源编码

资源	级别	类型	版本号	资源 ID
智能卡读取器	112	见 F8.3.2.1	1	00700041

表 F110 智能卡 apdu _ tag

apdu _ tag	标记值	资源	方向（主机<——>应用）
T smart _ card _ cmd	9F 8E 00	智能卡读取器	<———
T smart _ card _ reply	9F 8E 01	智能卡读取器	———>
T smart _ card _ send	9F 8E 02	智能卡读取器	<———
T smart _ card _ rcv	9F 8E 03	智能卡读取器	———>

资源类型编码

智能卡读取器资源可以有至多 16 个实例，可以由主机或其他模块提供。应用通过在资源类型域最低 4 比特明确所需的设备号来选择所需的读写器，如图 F38 所示。

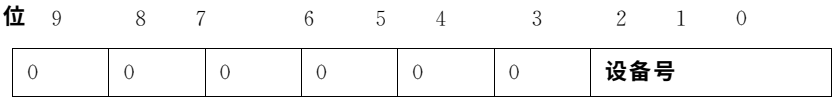


图 F38 智能卡读取器资源结构

F8.4 EPG 将来事件支持类

这个资源由 EPG 应用提供，以支持 CA 模块和 EPG 之间的授权信息的通信—可以基于主机或模块实现。为 EPG 应用提供授权信息的 CA 模块，应和资源管理器所列出的所有资源建立会话连接，并按照 EPG 应用所要求的协议操作。

这个协议使用两个对象：一个 event _ enquiry 事件查询对象和一个 event _ reply 事件响应对象。这个协议允许 EPG 根据 SI 信息中的事件 ID 来查询当前或将来事件的授权状态。CA 模块可以回答授权情况为：可用、不可用、在对话后可能可用（例如付费收看等）、授权状态未知等。而且协议允许 CA 模块开始一个对话过程，来使得潜在的授权变为可能，例如付费收看方式，然后在返回到 EPG 对话。本资源并不阻止同一或合作的厂商使用专用的资源机制来使得 EPG、CA 的操作更加完美。如果 CA 模块向独立的 EPG 提供授权信息，将使用本资源。

F8.4.1 对象

F8.4.1.1 事件查询对象

这是从 EPG 到 CA 模块的查询对象。见表 F111。

表 F111 事件查询对象编码

句法	比特数	类型
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

见表 F112。

表 F112 event _ cmd _ id 取值

event _ cmd _ id	取值
mmi	02
Query	03
Reserved	其他值

“MMI”命令允许 CA 模块启动 MMI 对话过程，以使得授权有效。典型地它在 EPG 已经形成（在此之前使用“查询”命令），并与 CA 模块建立会话之后发送。这样可以暂时关闭自己和 MMI 会话，使得 CA 模块可以使用相应的 MMI 资源。“查询”命令用于请求当前的事件的授权状态，而不需要请求一个对话过程。SI 中的事件 ID 作为唯一的标志，对其他所有的位置标志符都有效。

F8.4.1.2 事件应答对象

这作为 CA 模块对 EPG 的应答。见表 F113。

表 F113 事件应答对象编码

句法	比特数	类型
event _ reply () {		
event _ reply _ tag	24	uimsbf
length _ field ()		
event _ status	8	uimsbf
}		

event _ status

见表 F114。

表 F114 event _ status 取值

event _ status	取值
entitlement _ unknown	00
entitlement _ available	01
entitlement _ not _ available	02
mmi _ dialogue _ required	03
mmi _ complete _ unknown	04
mmi _ complete _ available	05
Mmi _ complete _ not _ available	06
Reserved	其他值

entitlement _ unknown 表示 CA 模块不能确定这个事件的授权状态；entitlement _ available 表示这个事件当前的授权可用；entitlement _ not _ available 表示这个事件的授权不可用，并且通过和 CA 模块对话也不能取得授权；mmi _ dialogue _ required 表示这个事件的当前授权不可用，但是通过用户和 CA 模块的对话可以取得授权；mmi _ complete _ unknown 在 CA MMI 对话结束但授权状态仍然未知的情况下，作为对使用 MMI 命令的事件查询对象的应答。

mmi _ complete _ available 在 CA MMI 对话结束并且授权已经取得的情况下，作为对使用 MMI 命令的事件查询对象的应答；mmi _ complete _ not _ available CA MMI 对话结束但授权未取得的情况下，作为对使用 MMI 命令的事件查询对象的应答，例如用户在对话过程中决定不购买业务。

注意事件应答 event _ reply 对象是作为对上一个接收到的事件查询 event _ enquiry 对象的应答，因此在 EPG 接收到当前查询的 event _ reply 之前，必须不能发送新的另一个 event _ enquiry 对象。

F8.4.1.3 使用举例

下面给出了典型的交换协议。这里将使用 MMI 资源的增强功能，以允许应用在返回到当前业务之前利用短暂的延迟来关闭 MMI 会话，以便另一个应用可以和前一个对话过程无缝地启动自己的对话过程。

EPG：事件查询 event _ query（query，event x）；

CA：事件应答 event _ reply（mmi _ dialogue _ required）；

EPG：关闭 MMIclose _ mmi（delay）、记录上下文；

EPG：事件查询 event _ query（mmi，event x）；

CA：打开 MMI open mmi、进行对话；

CA：关闭 MMI close _ mmi（delay）；

CA：事件应答 event _ reply（mmi _ complete _ available）（可回答“不可用”或“未知”）；

EPG：打开 MMI open mmi、根据记录的上下文继续 EPG 对话。

F8.4.2 EPG 将来事件支持资源编码

见表 F115、F116。

表 F115 EPG 将来事件支持资源编码

资源	级别	类型	版本号	资源 ID
EPG 将来事件支持	120	见 F8.4.2.1	1	00780041

表 F116 EPG 将来事件 apdu _ tag

Apdu _ tag	标记值（HEX）	资源	方向（EPG<－－>CA）
T event _ enquiry	9F 8F 00	EPG 将来事件支持	－－－>
T event _ reply	9F 8F 01	EPG 将来事件支持	<－－－

资源类型编码

一个主机上可能有多个 EPG 存在。例如主机可以支持一个内部 EPG 应用，而用户购买了一个基于模块的 EPG。这种情况下，主机从 0 开始分配资源类型号，以区别运行的不同 EPG 实例。支持 EPG 的 CA 模块使用这个资源时，应和资源管理器作为不同资源类型进行广告的资源的每个实例建立一个会话。

附 录 G
（提示的附录）
多密技术的使用

本附录主要有两部分内容：一是解释为什么公共接口要设计成附录 F 这样；二是说明如何实现和使用公共接口，它包括对附录 F 中未限定的各种设计选项的建议。

本附录同时也对附录 F 实现各种应用时，扩充公共接口提出了建议。这些建议最大可能地保证模块和主机之间的完全互操作。设计者是否采纳这些建议可自由选择，但是如果设计者不采纳这些建议的话，设计者应确信自己充分理解这样做的含义及其可能对产品互操作性带来的影响。

G1 概述

设计附录 F 首先是为了满足条件接收的要求，然而，CA 只是数字电视接收机所需的应用之一，设计一个能支持其他应用的通用接口比只为 CA 设计的接口用处会大得多。通过在主机与接口另一侧的应用之间适当地划分功能，各种有关的应用就能清晰的隔离开来。

举例而言，假定按目前大多数 CA 系统实现中所做的那样，以某种方式在主机和基于模块的应用之间划分了条件接收功能，则接下来需要作的大量工作就是定义驻留在主机中的通用 CA 功能与驻留在模块中的特定 CA 功能的相互操作，同时当今所有的 CA 系统供应商，或许还有一些潜在的 CA 系统供应商和主机生产商都需要同意这些定义，而这是不现实的。

因此，需要通过接口传输全部 MPEG 传送流。原则上只要同意使用统一的加扰算法就可以通过在主机上实现通用解扰器而节省成本。然而解扰器是构成任何条件接收系统安全性的一个不可分割的部件，由于这种安全性方面的考虑和大家不愿就公共解扰器控制接口达成一致，致使在模块上放置解扰器成了系统的一个基本特征。这是主要的原因，同时它还可以支持更多可能的应用。公共接口因而成为通用 MPEG 流输入输出端口，它包含所有必要的功能部件以容纳未来的功能。

命令接口有 5 层，其中一层有两个子层。它允许以下列方式划分所有的接口功能，即允许优化每层的全部接口功能而与其他各层几乎无关。这种方法的优点之一是：当在一个物理层的连接上建立多个传输层连接时，无需实际改变设计，并且会话层的使用以及资源的充分发展都不影响传输层或其他下面的层。

G2 物理层

G2.1 传送流接口

传送流接口是双向 8 位并行的，在输入输出方向上分别有两路控制信号和各自的字节时钟。在每个接口上有以字节时钟速率传输的连续字节流，被分成 188 字节为单位的 MPEG-2 传输包，并且当传输无效的数据时在字节流间可以存在空隙。MxSTRT 信号（x＝I 或 O，取决于传送流的方向）有效时间为一个字节，指明每个包的初始（同步 sync）字节，MxVAL 指明是有效字节。取决于主机的实现，MxVAL 的操作有 3 种可能性：

a）传输包依序传送，相互之间根本没有空隙，这种情形 MxVAL 恒为有效；

b）传输包以 188 字节组为单位传输，前后组之间有无效字节间隙。在这种情况中 MxVAL 在 188

字节期间是有效的，接下来一段时间无效直到下一传输包变成有效；

c）无效字节既可以在一个传输包中出现也可以在传输包之间出现，MxVAL 信号根据需要上下跳变，但变化规律无法预测。

情况 a）和 b）在实际主机中可能是常见的，然而也应该允许情况 c）存在的可能性，除非在主机制造商之中有共同的协定，即现在或在未来设计中都不会出现此种情况。

传输码流接口给主机和模块设计者都提供了相当大的自由度，将会带来不可忽略的 PCR 抖动，取决于主机和模块的实现。主机可通过给模块发送特性良好的流来控制模块产生的抖动，模块也返回一个类似的质量良好的传送流，若主机发给模块的流之间包含大量宽度变化的间隙则模块可能返回明显增加了 PCR 抖动的类似码流。规范通过间接规定返回码流中 PCR 抖动与输入码流中间隙数目的函数关系来实现这种特性。

附录 F3.4 中“数据和命令逻辑连接”规则 3 规定：

当处理一输入传输包时模块应引入一段恒定的延迟，加到每个字节的最大延迟变化（tmdv）由下列公式决定。

为避免混淆，这里的意思是：

在该句子中，每个字节在 MPEG 传输包中的位置应由 i 表示，加到字节 i 的最大延迟变化（tmdv）由下列公式计算：

$tmdv_{max} = (n * TMCLKI) + (2 * TMCLKO)$

每个字节 i 的延迟将取决于模块的输入和输出时钟周期（TMCLKI 和 TMCLKO）、n 和 i。因此，在 MPEG 包中的每个字节可能有不同的时延，然而，这种时延上的抖动（延迟变化）与 i 无关。

G2.1.1 时钟信号

PC 卡的电气规定主要为存储器或低速的 I/O 卡而考虑。因此，它没考虑用于公共接口的快速时钟信号。本附录描述这些信号的规定。

G2.1.1.1 设计原理

出于实际使用的考虑（例如为使用标准的 HCMOS 器件和使电磁兼容设计更为容易）需要放慢上升和下降时间。因为时钟通过许多串联的模块会有潜在的累积的失真，不是在模块卡中就是在主机中，有必要提供某些形式的时钟再生。出于经济性考虑，应该允许主机采用简单的时钟缓冲解决方案，虽然这会带来不可避免的失真。

在这些情形下，很难保证应用到最高允许频率而没有缺陷，这里指将不能保证互操作性。合理的设计应符合图 G1：

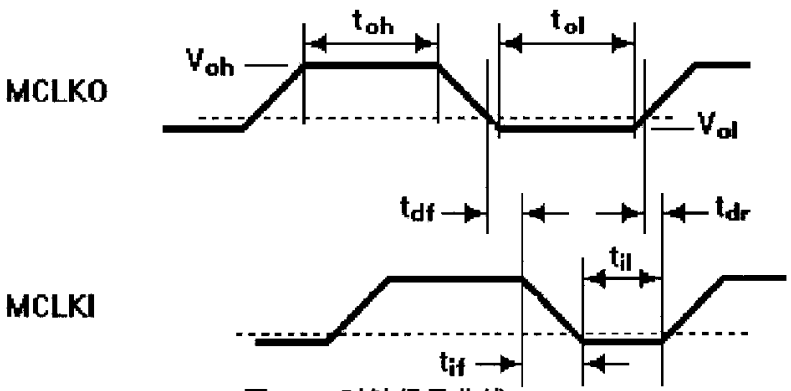


图 G1 时钟信号曲线

使用的符号定义如下：

t_{ol} MCLKO（模块时钟输出）输出的低电平时段；

t_{oh} MCLKO 上输出的高电平时段；

t_{of} MCLKO 输出的下降沿时间；

t_{df} 下降沿通过主机缓冲的传输延迟；

t_{dr} 上升沿通过主机缓冲的传输延迟；

t_{il} MCLKI（主机时钟输出）输出的低电平时段；

t_{ih} MCLKI 输出的高电平时段；

t_{if} MCLKI 输出的下降沿。

根据主机上检测器件的门限范围（这里必须忽略噪音电压范围），穿越门限的下降沿和上升沿的时间间隔可低到 $t_{ol}=40$ ns，缓冲因它的不对称性可能将此参数减少 $t_{df}-t_{dr}$（下降沿可能比上升沿多延迟一个上升沿时段），而将有它自己的上升和下降时间，然后便能算出对于模块卡而言是有保证的高低电平时间间隔。

第二模块输入端低电平时段的最坏情形是 $t_{il}>=t_{ol}-t_{df}+t_{dr}-t_{if}$ 为低电平时段，$t_{ih}>=t_{oh}-t_{dr}+t_{df}-t_{ir}$ 为高电平时段。

G2.1.1.2 使用指南

时钟信号失真

当前的 AC 规范要求时钟的高低脉冲宽度最小 40 ns，周期 111 ns。如果一个模块或主机使用这个时钟作为输入去产生一个缓冲输出，要确保输出时钟不至于失真到不再合乎时钟规范的要求，但这是十分困难的。

在任何情况下，只要提供给模块的输入时钟信号 MCLKI 符合下面“时钟上升和下降时间”下给出的规定，模块应输出符合 AC 规范的时钟信号 MCLKO（高低电平时段分别在 Voh＝2.4 V 和 Vol＝0.5 V下测量）。如有必要可对 MCLKI 规范作适应性修改，以便实现级连的 CI 模块。

时钟上升和下降时间

所有陈述在上升和下降沿上都适用。主机和模块之间电平的不对称性是 PC 卡规范的反映。

模块时钟输出 MCLKO：

— V_{ol} 是 0.5 V，V_{oh} 是 2.4 V；

— 在 V_{ol} 和 V_{oh} 之间单调升或降；

— 处于低电平（V_{ol}）和处于高电平（V_{oh}）的最小时段均是 40 ns；

— V_{ol} 和 V_{oh} 之间最大的转换时间是 20 ns；

— V_{ol} 和 V_{oh} 之间最小的转换时间是 5 ns（以尽量减少波动冲击和电磁干扰 EMI）。

注意，如果模块可能输出规定的最小时钟时期（111 ns），这些约束限制就特别难以达到，因为转换时间必须变得更少（但任何负载条件下都不小于 5 ns），脉冲周期占空比保持平稳不变。因此，模块通过简单地整形输入的时钟信号 MCLKI 来产生 MCLKO 被看成是特别困难的甚至可能是不切实际的。将规范中 MCLKO 的高低电平时间段减小到少于 40 ns 会使对主机缓冲电路的要求变得过于苛刻。

主机时钟输出 MCLKI：

— V_{ol} 是 0.5 V，V_{oh} 是 2.8 V；

— 在 V_{ol} 和 V_{oh} 之间单调升或降（Monotonic）；

— 处于低电平（V_{ol}）和处于高电平（V_{oh}）的最小时间段均是 20 ns；

— V_{ol} 和 V_{oh} 之间最大的转换时间是 20 ns；

— V_{ol} 和 V_{oh} 之间最小的转换时间是 5 ns（以尽量减少波动冲击和电磁干扰 EMI）。

注意对于将一模块卡的时钟缓冲后传送给下一个的主机而言，有一个限制条件，那就是：缓冲的不对称性（上升沿和下降沿通过缓冲的传播时间的绝对差值）和其输出的转换时间之和不应超过 20 ns（由上面第二段给出）。

时钟信号电容

所有的设备将遵守以下最大电容规范：

当前模块输出时钟 MCLKO 的最大电容负载是 50 pF，最小 10 pF。—当前主机输出时钟 MCLKI 的最大电容负载是 25 pF，最小 5 pF。

下过冲和上过冲

在所有时间对所有信号：

$-0.5\ V \leqslant Vo \leqslant Vcc + 0.5\ V$

G2.2 命令接口

在主机方面，接口已由附录 F 规定，数据应如何发送到模块和从模块方发出是确定的。附录 F 除了说明其所做的一切操作必须使主机方以规定方式动作外，并没有规定模块应如何操作它自己这边的接口。因而，接口的模块卡这边可准确地镜像主机所用的面向字节的数据传输，或者更直接地可以允许寻址访问传输缓冲区，模块也可以象主机一样工作在查询方式，或者可在中断驱动方式下运行。这些是由接口硬件设计者自由选择的。

在物理接口上，除了复位（在命令寄存器中设置 RS 位）需要一点时间才完成，通过设置状态寄存器中的 FR 位来告知完成外，假定所有的命令立即生效。

G2.2.1 避免死锁

模块设计者必须小心以避免在物理层协议的操作中发生死锁。对那些在主机和模块之间各个方向的数据传输使用各自缓冲区的设计来说这不是主要问题，然而低成本的模块解决方案可能只使用单个缓冲区。在这种情况下当缓冲区为空时，主机和模块必须仲裁以获得缓冲区访问权，并且在缓冲区被使用后重新获得访问访问权。模块设计者必须保证模块与依照规范运行的一台主机通信时，他的设计实现是无死锁的。

G2.2.2 速度

附录 F 要求命令接口物理层应该在每个方向支持至少 3.5 Mbps 数据吞吐量。这是对所选的任何物理层的基本要求，显然所选定的特定的 PC 卡规范符合这一要求。这不意味着所有的“实现”必须满足这个要求。数据传输的实际速率主要取决于物理层查询间隔和协商好的缓冲区大小的组合。模块或许仅有 16 字节缓冲区，不能期望这样的模块将能高速发送或接收数据。另一方面，主机必须有至少 256 字节缓冲区，并且必须设计成能处理相应的数据速率。物理层查询速率的一个合理目标值应为 10 毫秒，如果每次查询都能处理一个输入包和一个输出包，这样通过接口的有效数据率在每个方向上能高达 204.8 kbps。

G2.2.3 中断

附录 F 中物理层是一个查询方式接口。然而，中断驱动接口方式效率高得多，特别是用于多个槽的接口中，模块实现是很直接的。建议所有的模块应该支持中断方式。建议的实现方式如下：DA 与 FR 应该通过在命令寄存器上的两个新的中断允许位，分别是 DAIE（位 7）和 FRIE（位 6），经过门电路组合到 IREQ＃线上。能正常处理中断的主机会使用这些机制，但查询方式的主机将仅仅设置中断允许命令位为 0。这样命令寄存器如图 G2 所示：

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

图 G2　命令寄存器

RS，SR，SW 和 HC 预留它们原来在公共接口规范描述的功能。

DAIE 置位时，对状态寄存器中 DA 位的任何置位操作也将置位 IREQ＃。

FRIE 置位时，对状态寄存器中 FR 位的任何置位操作也将置位 IREQ＃。

其用法是直接明了的。当 DA 或 FR 被置位并且相应的中断允许位被设置时，IREQ＃被置位，随后调用中断子程序检查这两个状态位，在这一点上它与查询子程序相差无几。即使看到了中断，主机在检测 FR 位时可能仍会发现 FR 为复位状态，这并不奇怪，因为在单缓冲区设计中模块可能在中断业务子程序处理中断之前又获取了空闲缓冲区的使用权。

G2.3　单插槽和多插槽的对比

对单插槽的支持简单而直接，已经编写了规范，对模块实现加以限定，以给主机设计带来最大自由度。规范的目的之一是通过总线方式将命令接口连到所有插槽以实现多插槽接口，只有 CE1＃由各插槽自己提供，传输码流接口以级连方式连接，当未插入模块卡时通过一简单的三态缓冲器旁路数据和时钟信号即可。

G2.4　数字电视广播增强型全功能 PC 卡插槽

由于不能使用简单的总线和级连方法，成本可能会很高。每个插槽将必须由一片控制芯片单独支持，该芯片同时集成了 PC 卡标准功能和各种不同的数字电视广播功能。

G2.5　扩展器

这是一个外部器件，它带有多个插槽以供多个模块插接，并带有可插入到主机插槽上的导引端子，允许几个模块与一台单插槽主机一起工作。物理层的设计要允许这种工作方式，同时传输层的设计也考虑了对这种方式的支持。

G3　链路层

在这层和接下来所有各层内的论述只与命令接口有关，传送流接口的分层在其他地方论述。然而，后面也有一些关于许多应用需要直接从 MPEG-2 传送流提取信息的讨论。

G3.1　设计原理

这里只提供一个简单的协议来完成两个任务。第一个是用来拆分和重新组合信息包，以使它适合物理层启动时协商达成的缓冲区大小。第二个用来多路复用来自所有传输包的数据片段，这些传输数据包正排队等待通过某特定插槽发送，这样做可为它们公正地划分可用带宽。

G3.2 协议实现指南

主机端的链路层程序通过操作物理接口来收发信息片段，这是比较容易编写的，因为这一接口已经很好地规范了，而模块端的程序则主要依赖于在模块端采用的物理接口的具体形式。对于必须同时编写主机端和模块端的链路层的设计者来说，使物理层的存取程序对链路层协议程序呈现公共的接口是有好处的，因为这样的话，两边就可以使用相同的协议代码。

为了实现协议的数据分片多路复用部分，每一个片段都包含产生这一片段的传输包的传输连接标志符。当传递一个包到链路层等候发送时，标志符必须随着传输包由传输层直接提供——链路层不能直接从传输包中提取，其中的原因在传输层的实现指南中有解释。链路层必须以下列方式排队传输包：为正在进行发送的每一个传输连接维护独立的队列，而且那些数据分片应该以一种循环的方式，从每个队列的头部数据包中提取。

对链路层一种可能的优化是选择缓冲区大小，使之与到来的传输包相匹配。它可以找出传输包的第一个片段，读取传输报头的长度区，因为传输报头总是很短，甚至可以被包含在一个最小的片段内。

G4 传输层

G4.1 设计原理

通过这一层提供的方法，一个模块中的应用或者资源可以连接到主机。这一层的出现使划分模块的功能更加容易，也使得主机能更容易地管理该功能。传输层连接只能是从模块到主机。它是单一主机连接一个或多个模块的体系结构。附录 F 不支持模块之间的直接传输层连接，不支持多个主机间的连接，也不支持主机间或不同主机的模块之间的传输层连接。这些需求将在其他规范或者将来的版本中解决。

然而，设计的传输层协议能够支持模块扩充器的概念，这意味着把一个本身含有多个插槽的中间设备插到主机的单个插槽上可以支持多个模块。请求传输连接和新传输连接（Req_T_C/New_T_C）处理允许中间单元通过识别这些对象来建立路由表，而且可以通过删除传输连接和删除传输连接回应（Delete_T_C/D_T_C_Reply）处理来删除它们。因为以上原因，也因为有些模块可能希望采用多个传输连接，所以，即使只有一个插槽的主机也必须能够支持多个传输连接。

这一层协议继续采用物理层所用的主/从模式，主机是主控方。协议的轮询功能使该方式得以实现，而且，这里有一个特别的交易处理允许模块在传输数据时发出它有信息需要传送的信号，然后主机再明确的请求它。这样做之后，主机就能分配一个适当的接收缓冲区，否则的话，因为数据之外所有应答只有几个字节大小，主机也许因此会为其分配不合适的缓冲区。在主机处理数个模块时，其中只有一个或两个可能会随时发送数据，这样有利于主机优化缓冲区分配策略。但是对于模块没有这种考虑，因为它们只为一个或两个传输连接业务，而且，至少在目前的版本，发送到模块的对象实际上被限制在大约 256 字节，加上一个或两个包头。

G4.2 实现指南

G4.2.1 最小实现

在模块端，传输层协议是回应性的，它只对从主机收到的传输层包作出反应。它将任何需要发送的传输包存入队列，直到主机请求数据包。然而，主机必须管理两个方向的数据流。它周期性的轮询所有的传输层连接以从模块请求数据或协议管理包。对任何特定的传输层连接，在进行任何数据流交换处理时，必须停止这种轮询，而且只有当两个方向上的所有数据和协议管理通信都停止时它才能重

新开始。如果不这样做，在给只支持小的数据分片的模块传输大的传输层包时，那么可能会形成这样一种局面：无用的轮询信息堆积在主机的发送队列里。

要求链路层将来自不同传输层连接的数据分段公平地复用到同一模块会产生一个潜在的传输协议层问题。当模块请求一个新的传输层连接（Req _ T _ C），那么应该先发送请求传输连接的新的传输连接（New _ T _ C）再为新的传输连接发送创建传输连接（Creat _ T _ C），这是很重要的。因为链路层的接口把创建传输连接（Creat _ T _ C）和新的传输连接（New _ T _ C）看作不同的传输层连接，如果按习惯做法，链路层就很有可能在新的传输连接（New _ T _ C）前发送创建连接（Creat _ T _ C）。正确的做法是为新的传输连接（New _ T _ C）和创建传输连接（Creat _ T _ C）两者给链路层传递现存的连接标识符，然后链路层会正确地给它们排序。当然，新的传输连接应该使用的是在创建传输连接（Creat _ T _ C）传输层包中的传输连接标识符而不是现存的那一个，这也说明了传输协议子程序采用来自于包的传输连接标识符的必要性，而不是采用链路层用来发送数据分片的那一个。

象在附录 F 中说明的一样，主机必须至少能够同时支持 16 个传输层连接，即使它可能只有单个公共接口插槽。这是必要的，因为任何一个模块都可能需要多个传输层连接，而且通过使用外部扩充器，即使一个单插槽的主机也有可能支持多个模块。

在传输层，主机和模块都必须有重新组装数据包的能力。这就是说，在把结果包传递给会话层之前，必须串接连续的 T-date-more 数据包和最后的 T-date-last 数据包，无论主机和模块都不必把从会话层收到的大数据包拆分成若干个传输数据包。提供这一能力可以在更低的层中用来支持 GB/T 16649，目前它并不是规范中的一部分，但以后也许会是。

G4.2.2 多重连接

预计模块将分成两个主要的种类—— 一类使用业务资源而另一类提供业务资源。然而，有些模块会兼有这两种功能。在这种情况下，建议提供资源和使用资源的应用程序应该工作在不同的传输层连接。因为使用同一会话层编号但发向模块中不同的目的端的会话数据包可能会在同一传输连接中横穿交叉，而这样做就绝对避免了这一可能性。然而，建议每一个模块最多使用两个传输连接，而同一模块中不同应用程序间的任何交互通过会话层经由主机完成，应该构造成支持最多两个连接。

G4.2.3 失败模式

主机和模块都应该对请求实施超时操作，这样它们就可以处理请求总是得不到回应的情况。因为主机总是期待至少一个状态回应，所以它将需要对所有发往模块的请求执行这一操作，而模块只需要对请求传输连接（Req _ T _ C）请求实施超时，也许还有删除传输连接（Delete _ T _ C）请求也要实施超时。

传输连接的数量是有限制的，所以主机和模块需要能够处理下面这种可能性：当有一个新的传输连接需要创建时，所有的传输连接都已经在使用了。

可能会出现协议错误，也就是说，因为主机或者模块中的实现有缺陷而收到意外的传输数据包。因为协议错误可能意味着这一连接再也不可用，所以处理这种情况的最简单的方法可能就是关掉这一传输连接。

虽然简单地提示用户模块 X 出错也许会有帮助，但因为这是消费类装置，所以悄无声息的处理失败也许是最好的。

G4.2.4 推荐的参数设定

对于任何发送给模块的信息，主机应该设置超时为 300 毫秒。对于请求传输连接（Req _ T _ C）和

删除传输连接（Delete _T _C），模块也应设置300毫秒的超时。

G5 会话层

G5.1 设计原理

这一层提供的机制使得请求能够得到它们要用的业务资源的访问权限。主机内的资源管理器收集和管理有效应用这一机制所需的信息。资源模型的想法很早在开发接口的讨论中就发展起来了，然而只是到后来的阶段，这一概念才被完整的设计出来，并和会话层结合在一起作为它的业务机制。资源概念是应用层具备可扩展性的关键，而且，会话层允许主机处理与另一模块提供的某资源的通信交换，而只需要知道它的标识符和提供这一资源的传输层连接。

G5.2 资源

在开始会话时，应用程序提供一个资源标识符作为请求的一部分。为了将已创建的会话和资源联系起来，主机上的会话层协议程序参考主机上的资源管理器创建的数据结构，该数据结构是会话层运行它自己的应用层协议的结果。资源管理器自己有一个大家都知道的资源标识符，所以模块上的应用能够和这一标识符建立会话，在使用资源之前可以得到可用资源的信息。

G5.3 实现指南

G5.3.1 建立和关闭会话

请求使用资源的应用发送“开始会话请求”对象到主机，它们将收到“开始会话应答”对象作为回答。通过用“建立会话请求”对象和“建立会话应答”对象，主机能将会话扩展到另一个模块。通过“关闭会话请求”对象和“关闭会话应答”对象，会话的每一方都可以结束会话。实现时要保证所有的请求—应答处理在会话开始任何应用层通信交换之前完成，这很重要。

所有会话标识符都由主机分配。由主机直接提供到资源的会话是一个直接的过程。然而，当涉及的资源由另一个模块所提供时，主机通过自己扩展会话连到提供者模块，这种情况下，有两种可能性。主机可以使用与两个传输连接所用的相同的会话标识符，或者它可以在每一个传输连接使用不同的会话标识符。第一种方法实现起来比较直接，但是第二种允许支持的会话总数更多。实际上，如果使用16比特的会话标识符，同时存在65 535个会话一般没有问题，而大多数实施者往往选择使用一个会话标识符。这样做对使用传输层连接的潜在影响已经讨论过了。然而，规范确实提供了两种选择，所以实施者需要注意，他们不能假定两个模块间经由主机的一个会话在两端必定会用相同的会话标识符。

G5.3.2 会话的生存期

一些资源，象资源管理器，预期会同时支持许多会话，它们中的每一个也许会持续比较长的时间。其他资源也许只是一次支持一个会话，而且预期它的活动时间很短。会话层没有以任何方法试图去管理会话的生存周期。在这方面它依赖应用程序是合理的。在这种情况下，“合理”是指直到需要资源时再创建会话，而且只要不再需要一个会话，要尽可能快的关闭它。

G5.3.3 失败模式

会话层应答对象携带会话层操作成功的状态信息或者其他信息。资源也许会不存在，不可用或者忙，会话层会通知以上信息。应用程序如何使用由它们自己决定。在一些情况下，它可以被忽略掉，而在其他情况，也许有必要提示用户出了问题和建议的改正动作。

如果一个传输连接中断，那么它所承担的所有的会话将会被主机终止。若会话在两个模块之间进

行，在剩存的传输连接上会话还有一半，会话也将被终止。

G5.3.4 推荐的参数设定

主机至少应该允许支持同时存在128个会话。

G6 应用层

G6.1 设计原理

在模块上运行的应用程序利用主机提供的服务以完成其功能。这些服务被划分成各种各样的资源，这些资源将所有能提供的功能包装成多个易于管理的部件。尽管从应用程序角度看，所有的资源由主机提供，但以上提供的机制能使有些资源由其他模块提供，而主机只充当代理，负责在提供者模块和使用者模块间转送资源请求和响应。本规范的基本设计目标是将资源定义为简单的底层功能，这样就可以或必须由主机提供应用程序的一些公共服务，而不必为此定义特别的应用。例如，条件接收应用程序负责由主机和模块共同提供的全部CA功能，主机提供（直接提供或代理方式）对于许多可能应用都通用的功能诸如用户接口、低速通信。特别为条件接收提供的资源只有一项，用于提供用户选择业务的信息和用户要求的交互状态（若有的话）以使这些业务可用，但这一资源并不专属于任何特定的CA应用，那些需要获取主机中业务选择状态的非CA应用也能使用该资源。提供资源的程序和使用资源的程序都能以模块化形式编写，这样资源间的相互调用有限且易于理解。每一资源提供少量的几个对象，这些对象定义一套专门用于该资源的协议，不必依据通用的应用程序分层协议模型的限制来运用资源，一旦建立了与资源的会话，只需使用公共接口规范或其附录中的资源描述就可完全定义应用层对象的通信。

G6.2 死锁

如果不仔细设计应用程序，无论何时有超过一个应用程序争用资源，就可能会发生死锁。当两个应用程序在相互等待对方释放资源才继续执行时，它们之间就出现了死锁。更复杂的死锁状况能在多个应用程序间发生。当一个应用程序需要多个资源以完成操作时，它应该在启动前申请所有需要的资源。如果申请过程中发现某资源正被占用，它应该释放所有申请到的资源，以后再试，而且再试运行之前的等待时长最好不固定，而是等待一段随机时长。这将有助于确保两个争用同一资源的应用程序延迟之后再运行时不大可能正好在同一时间再做同样的操作。

G7 资源

G7.1 最小的资源

在附录F中描述的所有资源是任何数字电视广播兼容的主机应该提供的最小资源。

G7.2 应答时间

这些应答时间应该包括规定的连接轮询最大间隔（100毫秒）。应答时间应该以下面这种方法测量：

—主机等待模块时间：

这一时间应该从主机收到回应原始对象的第一个确定的R_TPDU算起，到主机收到包含模块响应的第一个R_TPDU；

—模块等待主机时间：

这一时间应该从R_TPDU发送开始算，到模块接收到包含主机响应的第一个C_TPDU；

—以下给出每一资源的两个时间：

推荐的资源最长应答时间：它应该是在正常操作的情况下，资源应答所需的最大时间；

超时时间：这是一个应用在发出超时和出错信号之前，需等待资源响应的最小时间。

G7.2.1 CA 支持

收到一个 CA _ PMT 后，模块最多可在 5 秒后开始解扰和响应这一由主机发送的 CA _ PMT，从而允许 CA 系统内部处理广播信息。

然而，模块应该尽可能快地响应，因为这直接影响频道改变的速度。

为响应一个 CA _ PMT（询问）会发出一个 CA _ PMT _ replay 对象，推荐设置这一应答对象的响应时间为 100 毫秒，超时为 500 毫秒。

如果在模块回应前一个 CA _ PMT 之前（如果它需要回应一个 CA _ PMT _ reply）收到另一个 CA _ PMT 对象，那么这个模块只需要响应最近一个 CA _ PMT。

G7.2.2 主机控制—调谐

从主机到调谐对象不需要显式的响应，然而当主机已经调到一个新的业务时，它需要发送 CA _ PMT 对象。在重新调谐的时段内，主机不能因为任何其他原因而发送 CA _ PMT，以保证这一 CA _ PMT 对象是对调谐对象有效的确认。

推荐最大响应时间：3 秒。

超时：10 秒。

G7.2.3 主机控制—要求释放

通过这一对象，主机能够从模块重新得到控制权。作为响应，模块应该关闭主机控制会话，而且如果必要的话，需要发送一个清除取代（Clear Replace）对象。

推荐最大响应时间：100 毫秒。

超时：500 毫秒。

G7.2.4 日期—时间查询

推荐最大响应时间：300 毫秒。

超时：1.5 秒。

G7.2.5 显示控制

推荐最大响应时间：300 毫秒。

超时：1.5 秒。

G7.2.6 键盘控制

推荐最大响应时间：100 毫秒。

超时：500 毫秒。

G7.2.7 Comms _ cmd

主机用一个通信应答指令（comms _ reply）对象响应通信指令（comms _ cmd）。

测量应答时间应该是以连接到主机内的低速通信资源的时间为准，而不是完成到低速通信媒体的远端的连接的时间。

推荐最大响应时间：300 毫秒。

超时：1.5 秒。

G7.2.8 Comms _ send

主机用一个通信应答指令（comms _ reply）对象响应通信发送指令（comms _ send）。

推荐最大响应时间：300 毫秒。

超时：1.5 秒。

G7.3 资源管理器

G7.3.1 服务资源的目的

资源管理器代表应用和其他资源管理主机所知道的所有资源。它有一个“众所周知”的资源标识符，而且它实现的协议收集所有资源的信息，并把这些信息提供给应用，如果必要，也提供给其他资源。

G7.3.2 资源的最终用户

所有应用程序和资源的提供者都会用到这一资源。

G7.3.3 此资源的提供者

它是专门由主机提供的。基于模块的资源提供者不能通过发布其新的版本来取代它。

G7.3.4 其他方的介入

此协议从任何的基于模块的资源提供者那里收集可用资源信息作为它的部分功能。

G7.3.5 资源管理器描述

资源管理器收集所有关于应用程序可用的资源的信息，无论是直接由主机提供或者由存在于模块上的资源提供者提供。当应用第一次创建一个到资源管理器的会话时，资源管理器会向应用提供这些信息。它也用于管理可用资源的变化通知。但注意这只是通知资源是否有效，并不是它是否正在被另一特定应用占用。出于完整性和规范性考虑，作为应答发送到应用的资源列表应该包括资源管理器本身。

G7.3.6 强制要求

G7.3.6.1 主机端

主机必须提供资源管理器资源。

G7.3.6.2 模块端

提供资源的模块必须与这一资源创建会话，并参与协议的资源信息收集部分和资源更新部分。

G7.3.7 推荐参数设定

资源管理器资源至少应该支持 16 个会话。

G7.4 应用程序信息

这一资源是应用程序用来给主机提供关于他们自己的信息的。然而，如果希望的话，更重要的作用是主机能进入一个应用程序呈现的目录树。为了进入应用目录，主机向应用发送一个“进入目录”（enter _ menu）对象。应用必须创建一个人机界面 MMI 会话并显示一些东西来作为响应。如果在应用程序没有目录结构的情况下，这可以是一个简单的信息显示，并且只停留一会儿或者要求一个按键动作的响应。否则，应用程序应能管理单个目录或者目录树，以让用户与应用程序相互交流。

推荐参数设定

应用信息资源至少应该支持 16 个会话。

G7.5 条件接收支持

G7.5.1 特性的目的

模块与用户的遥控器没有直接的相互交流。只有主机有这种交流，从而知道用户选择了哪一个节目。主机一得到这一信息，必须立即通过发送节目号（programme _ number）通知模块：这是 CA _ PMT

和 CA _ PMT _ reply 对象的主要目的。

为了避免模块解码 MPEG2 PMT 表（主机已经完成了这一操作），主机还给模块发送一个 PMT 摘要，它包括了模块用来解码选定节目的所有的信息。

当用户同时选择多个节目时，节目列表管理的特色也被补充进来了。主机也许会改变整个列表，或在现存的列表中添加一个新的节目，或者当列表结构发生变化时，更新这一列表。

最后，因为主机上连接有多个模块，所以在解码开始之前，要执行一个程序以允许主机了解哪一个能最合适地解码所选的节目。当至少有两个模块可以解码一个即时按次计费模式的节目，在这种情况下，只有用户能通过一个合适的对话框决定使用哪一个模块。

G7.5.2 特性的最终用户

在模块上运行的一个 CA 应用。

G7.5.3 特性的提供者

主机的 CA 支持资源。

G7.5.4 其他方的介入

通过主机或者模块提供的合适的对话框，选择并且在两个或更多的侯选节目中作出决定的用户。

G7.5.5 特性的描述

在应用这一特性之前，必须由运行在模块里的 CA 应用，通过主机提供的 CA 支持资源，创建一个会话。它的标识符是“0x00030041”。

紧接创建会话之后的通常是一个从主机到模块的 ca _ info _ enq 和一个从模块到主机的 ca _ info。这一交换是重要的，因为通过选择拥有相符的 CA _ system _ id 的模块，它允许主机实现对模块的第一层过滤。虽然这一过滤是可选的，但我们建议主机制造商实现它。

现在主机准备好，可以使用这一特性了。这里有两个对象：从主机到模块的 CA _ PMT 和从模块到主机的 CA _ PMT _ reply。

G7.5.5.1 CA _ PMT

当用户选择一个非加扰节目时（如：一个 PMT 里没有 CA 描述符（CA _ descriptor）的节目），主机发送 CA _ PMT。这种情况下，主机应该发送一个没有任何 CA 描述符（CA _ descriptor）的 CA _ PMT（即一个节目信息长度＝0 并且 ES _ info _ length＝0 的 CA _ PMT）。

发生下列事件时，主机向模块发送 CA _ PMT：

a）当 CA 应用在模块中时，创建了一个有 CA 支持资源的新的会话，并且用户选择了一个节目；

b）用户或者一个“调整”命令改变了所选节目的列表，或者改变了已经选定的节目的已选基本流的列表；

c）PMT 发生改变并且关系到至少一个的所选节目中。这一改变由版本号（version _ number）的改变或者 PMT 表中的 current _ next _ indicator 的改变来告知。主机不知道这一改变是否影响 CA 操作。它只是警告负责检查这一改变是否对 CA 操作有影响的 CA 应用。只要有可能，这种变化对用户来说就应该是不可见的（当 CA 应用与对象交涉时，它继续自己的解码）。

通过移走除了 CA 描述符（它包含了 ECM 的 PID）外的所有的相对应的 PMT 描述符，主机为一个选定的节目建立 CA _ PMT 对象。主机移走 CA 描述符之外的所有描述符是非常重要的，否则，模块可能不能正确应答。如果正在使用同密技术，那么 PMT 中的一个节目的记录也许包含多个 CA ID 的

CA 描述符。这种情况下，CA _ PMT 对象将会包含所有正在使用的 CA 描述符，这就是说，接收者将不会试图以一个特定的 CA 应用所通告的 CA ID 为基础作出选择。在这种情况下，模块将不会收到一个可以理解的 CA _ PMT，它应该丢弃这一对象。如果重复的错误一再出现的话，它也许会关闭这一 CA 支持的会话，然后重新创建它。

接收到一个 CA _ PMT 后，该 CA 应用只选择有一个 CA 描述符的基本流进行解码。

主机还要加上两个字节：ca _ pmt 列表管理（ca _ pmt _ list _ management）的字节和 ca _ pmt _ cmd _ id 字节。

ca _ pmt _ list 管理（ca _ pmt _ list _ management）允许管理选定节目的列表。事实上，要给每一个选定的节目发送一个 CA _ PMT 对象。ca _ pmt 列表管理（ca _ pmt _ list _ management）的规则如下：

a）如果选择了一个单一的节目，ca _ pmt 列表管理（ca _ pmt _ list _ management）==only。此 CA _ PMT 取代所有以前的程序选择；

b）如果选择了多于一个节目，就要发送多个 CA _ PMT。发送的列表的第一项带有 ca _ pmt 列表管理（ca _ pmt _ list _ management）==first（这会抹掉以前所有的节目选择）。而最后一项带有 ca _ pmt 列表管理（ca _ pmt _ list _ management）==last。其他中间项带有 ca _ pmt 列表管理（ca _ pmt _ list _ management）==more；

c）如果主机只是想添加一个新的节目，它简单的发送一个 ca _ pmt 列表管理（ca _ pmt _ list _ management）==add 的 CA _ PMT。这种情况下，所有先前选择的节目都被预留；

d）如果主机想更新列表中一个节目的 CA _ PMT，它发送一个 ca _ pmt 列表管理（ca _ pmt _ list _ management）==update 的 CA _ PMT。这种情况发生在主机探测到版本号（version _ number）或者 PMT 的 current _ next _ indicator 发生改变时。然后，模块中的 CA 应用检查这一改变是否影响到 CA 操作。当然在一个选定节目的基本流的列表发生改变时（举例来说，用户选择了另一种语言），这种情况也会出现。这种情况下，主机必须重新发送更新后节目的整个基本流列表。

当多于一个模块有潜在能力解扰选定的节目时，ca _ pmt _ cmd _ id 被创建用来正确处理模块选择节目。如果只连接了一个带有正确 CA _ system _ id 的模块，就发送一个带有 ca _ pmt _ cmd _ id=ok _ descrambling 的 CA _ PMT。其他情况下，用其他的值："query" 代表此模块不能开始解扰或 mmi 对话；"ok _ mmi" 代表模块可以开始 mmi 对话，但不能开始解扰；在 "query" 或 "ok _ mmi" 之后被发送的 "not _ selected" 是为了告诉此 CA 应用已经选择了另一个 CA 应用。当收到 "not _ selected"，此 CA 应用适当地关闭可能已经开始了的 mmi 对话。选择节目的描述在附带的流程图中具体给出。

G7.5.5.2 CA _ PMT _ reply

CA _ PMT _ reply 负责应答运行在模块中的 CA 应用。只有应答带有 ca _ pmt _ cmd _ id=query 或者 ok _ MMI 的 CA _ PMT 时，它才会被发送。应答一个带有 ca _ pmt _ cmd _ id=ok _ descrambling 的 CA _ PMT 时，它不会被发送。CA _ PMT _ reply 用在模块选择节目中。选择节目的完整的描述在附带的流程图中给出。CA _ PMT _ reply 包含一个 CA _ enable 的参数，它指出，对于整个选定节目或者选定节目的每一个基本流，解扰是否可能，而且在哪种情况下。

对于同一程序的两个 ES，CA _ enable 也许会有不同的值。

G7.5.6 操作规则

G7.5.6.1 一般规则

—如果模块同时收到另一个 CA _ PMT 从列表中删除节目 x，那么它也许不会对一个 CA _ PMT（节目 x，“query”）或一个 CA _ PMT（节目 x，“k _ mmi”）作出应答；

—模块必须返回一个带有位于同一级别的节目或基本流 CA _ enable 参数的 CA _ PMT 应答（CA _ PMT _ reply）——作为进来的 CA _ PMT 的 ca _ pmt _ cmd _ id 参数；

—当收到一个包含几个参数的 CA _ PMT，ca _ pmt _ cmd _ id＝“k _ mmi”，此 CA 应用应该启动尽可能少的 MMI 对话（有代表性的，例如，当它发现实际上所有的基本流共享相同的 ECM 时，一个对话就足够了）；

—当 CA 应用收到一个带有的新节目的列表的 CA _ PMT 时，它开始分析这一新的列表并停止解扰前一个列表的节目，除非它探测到新的列表包含前面列表的节目或基本流。这种情况下，它应该不打断用户的观看继续解扰这一节目或基本流。例如，从表中移走一个节目的唯一途径在于发送没有这一节目的新的列表。这种情况下，模块应该没有中断的继续解扰列表中其他的节目。另一个典型的例子是当因为 PMT 的版本号（version _ number）发生改变时，主机发送一个新的 CA _ PMT。这一改变也许不会影响到 CA 操作，那样的话，模块不应该中断解扰；

—即使当用户（PMT 中没有 CA 描述符）选择一个未加扰节目时，主机也必须发送一个 CA _ PMT。例如：当用户选择一个已加扰的节目，然后选择一个未加扰节目时，为了取消选择并停止前一个节目的解扰，一个新的没有 CA 描述符的 CA _ PMT 被发送到 CA 应用；

—如果到一个 CA 支持资源的会话关闭，此 CA 应用应该试图开始另一个会话；

—对用户选择的任何当前节目当前都无关的 CA 应用也许会为用户对话创建 MMI 会话，例如：用户以前购买的另一个节目的临近的 PPV 事件的警告；

—当 PMT 版本改变，或者用户选择了没有 CA 描述符的程序时，主机应该发送一个不包含 CA 描述符的 CA _ PMT 对象。这一个空的 CA _ PMT 的在语意上与一个带有“没有选择（not selected）”的 ca _ pmt _ cm _ id 相同。

G7.5.6.2 加扰和未加扰的相互转换

— 当一个节目从加扰转换到未加扰时，有这几种可能性：

1） 这一改变只在数据包报文的 TSC 域或者在 PES 报文的 PES _ SC 域被告知，但在 PMT 中没有。这种情况下，主机没有理由发送一个新的 CA _ PMT 来移去列表中的节目。节目仍然保持被选择的状态，而且当 PMT 的版本号发展时，主机继续发送 CA _ PMT；

2） 这一改变导致 PMT 的修改。这种情况下，主机发送一个 CA _ PMT；

— 当一个节目从未加扰转换到加扰，有以下几种可能性：

1） 这一改变只在数据包报文的 TSC 域或者在 PES 报文的 PES _ SC 域被告知，但在 PMT 中没有。这种情况下，主机不会发送一个新的 CA _ PMT。必须由 CA 应用检测这一转换；

2） 这一改变导致 PMT 的修改（举例来说：CA 描述符被移走）。这种情况下，主机发送一个 CA _ PMT。

在这两种情况下，推荐 CA 应用尝试创建一个用户对话来通知用户。

G7.5.6.3 当所有基本流的 CA _ enable 不尽相同时，主机的行为

CA 应用也许会用包含不同 CA _ enable 值的 CA _ PMT（CA _ PMT _ reply）应答来回答。例如：CA 应用包含视频和音频的授权但是没有图文电视广播的授权。这种情况下，CA 应用这样应答：对于音频和视频基本流，ca _ enable＝＝“可以解扰”，但对于图文电视广播基本流，ca _ enable＝＝“不可

以解扰”。这种情况下，主机可以为音频和视频选择这一个CA应用，而且为文字电视广播基本流选择另一个有图文电视广播授权CA应用。

所以，CA应用解扰的选择可以由主机分别为每一个基本流完成。当这种情况发生时，主机应该能够以这种方式操作，但是这并不是强制的。主机并不需要把同一节目的几个基本流的解扰分开在几个模块中。当收到的CA_PMT应答有至少两个基本流有不同的ca_enable值时，对节目的所有基本流，主机应该应用一个最小的ca_enable值，这些值是由CA应用为程序中的每一个基本流返回的。当有几个CA应用同时连接在主机时，它应该立即选择一个，并用ca_enable==“01”为所有的基本流回答，否则它应该等待所有的响应并且选择给出最好回应的CA应用，这指的是能够解扰且不用调节基本流的最大数目的CA应用。但这最后也许会导致选择的CA应用不能解扰选定程序的所有的基本流。这种情况下，我们推荐主机给用户显示一条消息，说明选择的CA应用不能解扰所选节目中的某个基本流。当至少有两个CA应用只能够解扰选定的节目的一部分时，主机也可以给用户显示一条消息，让用户选择使用哪一个。

G7.5.7 强制要求

G7.5.7.1 主机端

主机必须提供CA支持资源。这一资源的所有指令的进程对主机来说是强制的。即使主机上只有一个接口（事实上，在这单个接口上连接扩展器以允许几个模块连接到主机是可能的），也必须由主机管理模块选择节目。

G7.5.7.2 模块端

模块必须创建一个与CA支持资源的会话。这一资源的所有指令的进程对模块来说是强制的。

G7.5.8 推荐参数设定

CA支持资源应该支持至少16个会话。

G7.6 主机控制

推荐参数设定

数字电视广播主机控制资源应该只支持一个会话。

G7.7 日期和时间

这一资源允许应用得到主机保存的时间。它也允许一个简单的，相对比较慢的警报器或基于时间的中断设备。根本的假定是主机维持它自己的时间定时器，此定时器与当前的多路传输的TDT表的信息同步。这一表的信息名义上是由UTC衍生而来的，而UTC有一个1秒的限定和一个未定义的错误，所以主机的时间观念是来自于UTC（和一个时区偏移量）。实际上，如果主机能够访问它认为比当前的多路传输的TDT更可靠的UTC信息源，那么它可以用这个信息源。这种情况下，日期和时间资源应该从刚才提到的那个信息源衍生。

虽然这一资源允许重复的日期时间对象有规律的发送到应用，但建议重复的频率应该限制在比1秒1次稍低。

推荐参数设定

日期和时间资源应该支持至少16个会话。

G7.8 人机界面

G7.8.1 概述

单个 MMI 资源支持两种模式的常驻模块的应用和用户间的相互作用：

a）高级；

b）低级（叠加部分屏幕或者全屏图像）

在高级模式下，应用能决定相互作用的内容，但是把相互作用的方法交给主机。例如：菜单对象允许应用定义一个选项列表。需要主机来把它显示给用户，并且与选定的应用（如果有的话）交流。在很多情况下，输出方式是通过主机 OSD 显示的字符，输入方式是通过在遥控器按键。然而，此规范不要求这种相互作用的方式。应用语言合成和声音辨认是现在设想的有效实现。模块应用应该被设计成这种有大范围的实现选择。特别地，回车和空格也许不会达到预期的结果。

当显示并把字符码或字符串放在显示器的特定位置时，低级模式用数字电视广播字幕作为描述位图的方法。这一 MMI 方法明确的假定输出方法是图像显示。输入方法允许有限的虚键码的字母表。

低级方法，与高级方法相反，主机没有参与应用的显示工程。还有几个主机不同的地方。下面是一些例子。

颜色分辨率（比特/像素）；

为图像提供的内存大小；

描绘速度；

所支持的 MMI 对象并发连接数；

可在应用之间移动的接口方法。

G7.8.2 特性的最终用户

人机界面的最终用户是模块中运行的一个应用。

G7.8.3 特性的提供者

人机界面的提供者是主机。

G7.8.4 强制要求

所有的主机应该支持至少一个 MMI 会话。这一会话可以是高级的也可以是低级的，这一点由应用自行决定。

G7.8.4.1 高级模式

当响应 enq、菜单或者列表对象时所有的主机应该能够与用户交流任何字符串的至少前 40 个字符，而且应该能容忍任何长度的字符串（也许通过切断它）。所有的主机应该能够与用户交流在 0 到 20 条的菜单或列表的用户菜单和列表对象。

在输出第一个 MENU _ more 或 LIST _ more 对象时，choice _ nb＝0xFF 和 item _ nb＝0xFF 允许应用不知道列表的长度。因此，主机需要在接收到指示列表结尾的“最后（last）”对象时，才计算菜单或列表条目。

在响应一个 enq 对象时，所有的主机都应该能支持长度达 40 个字符的条目。

如果在响应一个 enq 对象时，用户试图键入多于主机能接受的字符（或者多于参数 answer _ text _ length），那么主机应该给用户一个明确的指示，说明他们的输入是不可接受的。主机没有方法可用来通知应用用户试图提供过多的输入。

G7.8.4.2 低级模式

所有的主机应该能够在以下任选一种低级模式下操作。

a）低级叠加图像；

b）低级全屏图像。

主机是否实现对象缓冲技术是可选的。

G7.8.4.3 输入设备

公共接口规范要求的表示键盘编码的输入方法见表 G1。

表 G1 键盘编码的输入方法

键盘编码	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
意义	0	1	2	3	4	5	6	7	8	9	菜单	ESC	⇒	⇐	⇑	⇓	BS	RC

G7.8.4.4 输出字符码

主机应该能够用在 GB 2312 或 GB 13000 中的可显示的字符码与用户交流。

G7.8.5 字符编码

基于字符的输入/输出发生在以下情况：

a）高级 MMI 输出；

b）低级图像显示模式中，在数字电视广播字幕数据中提交的字符与字符串对象输出；

c）通过按键对象的字符输入；

d）通过 enq 和 answ 对象的字符串输入。

G7.8.5.1 字符表格选择

参照 GY/Z 174。

G7.8.5.2 多字节字符代码

参照 GY/Z 174。

G7.8.5.3 经由数字电视广播字幕（Subtitling）的字符

参照 GY/Z 174。

G7.8.5.4 高级模式输入字符代码

参照 GY/Z 174。

G7.8.5.5 低级模式输入字符代码

参照 GY/Z 174。

G7.8.6 文字排列方向

应用提交给主机的文字的排列顺序和方向也是用户习惯看到的顺序。

G7.8.7 输入设备设计指南

低级模式输入支持的代码组见表 G2。

表 G2 低级模式输入支持的代码组

键盘码	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
意义	0	1	2	3	4	5	6	7	8	9	菜单	ESC	⇒	⇐	⇑	⇓	BS	RC

为了允许应用的图像和文档符合这些输入代码，主机所用的视觉表示应该与上表中“意义”行的表示相同。当主机使用一个不同的表示时，它的用户文档应该包括一个描述，用来说明它的表示和

应用预期的上面的表示怎样联系起来。

在低级输入的应用中，有些键盘码有几个意义。它们的描述见表G3。为低级输入的应用产生这些键盘码的方法应该与为高级接口达到相同的功能所用的方法相同，也应该与主机自己的用户接口所用的相同。

表G3 键盘特定代码的含义

键盘代码	“意义”	使用	行为
A	菜单		待定
B	ESC	退出应用	这一个键盘代码不能被应用使用。它是一个给主机的信息，用户希望终止他们与当前应用的会话。 当在低级模式下按下此键时，主机应该执行下列次序的动作： （可选的）给应用发送一个载有ESC键码的key _ press对象。 给应用发送一个close _ mmi对象。 如果在适当的超时时间里（例如250毫秒），应用没有关闭这个mmi会话，主机用关闭会话要求（close _ seession _ request）会话协议对象关闭这一会话。
10	BS	取消	应用应该放弃这个用户界面屏幕，然后回到下一个更高的它的菜单结构。在这个用户界面屏幕恢复之前，应用保持一般状态。 BS键不应该终止应用。一般的，肯定应该有一个带有RC键的“退出应用”的提示。
11	RC	缺省，确定，选择	这一键码指出用户确实知道有当前的用户界面屏幕建议的动作。也就是说，他们已经在菜单选择了一个项目，来确认他们希望按这一选择行动。

G7.8.8 推荐参数设定

MMI资源应该支持至少一个会话。然而主机设计者可自由实现解决方案来允许向用户适当地显示多个MMI会话。

G7.9 低速通信

G7.9.1 简介

主机的一部分是一个低速通信资源类的接口。这将会通过例如电话线或回传电缆网络提供一个双向通信。它可以用来支持条件接收功能，并协助交互业务。

此资源类以一种普通的方式定义，这样可以用共同的方法使用不同的下层通信技术。该类中的资源类型被定义成支持特定设备类型使用公共的对象集通信。模型是一个同时的双向信道（全双工），通过它，任意的数据传输都是可能的。在应用和主机之间，双向都应用了流程控制。为了限制在应用端和主机端所需的缓冲区的大小，数据被分割成节来传输。流程控制协议也限制了在应用端和主机端的缓冲数目。

G7.9.2 特性的最终使用者

低速通信的最终使用者为模块中运行的应用程序。

G7.9.3 特性的提供者

低速通信的提供者为主机。

G7.9.4 特性的描述

在使用低速通信资源进行任何传输之前，必须由模块中的应用创建一个与此资源的会话。低速通信标识符是 0060xxx1 其中 xxx 是资源类型。有三种类型的设备可用：调制解调器、双向的电缆信道和串行端口。

定义的四个对象——通信指令（Comms Cmd），通信应答指令（Comms Reply），通信发送指令（Comms Send）和通信接收指令（Comms Rcv）。通信指令（Comms Cmd）由应用发送并且允许在通信资源执行几个管理操作。应答指令（Comms Reply）由主机发送并且答复通信指令（Comms Cmd）。它也答复通信发送指令（Comms Send），操作通信资源的外在的流程控制。通信发送指令（Comms Send）是发送到通信资源的一个数据缓冲。这是被送往队列的数据。通信接收指令（Comms Rcv）是一个从通信资源接收的数据缓冲。这是从队列中得到的数据。先前，缓冲区大小和超时值已经有通信指令（Comms Cmd）对象设定，然后应用到收到的缓冲。

为了使用一个特定的通信资源，必须用标准的会话创建机制创建一个与它的会话。然后，使用 comms 对象的传输会话取代这一会话。

G7.9.5 信道的连接

与信道的连接只限于调制解调器和双向电缆信道资源。当资源是串行端口时，没有理由建立一个连接：应用通过设置通信参数可以立即开始传输。

当开始一个调制解调器资源类型的会话时，应用应该通过发送指令“连接到信道”（Connect _ on _ Channel）把调制解调器连接到一个特定的信道。这需要提供一个电话号码描述符。如果是双向电缆信道，就假定主机“知道”怎样把电缆连接起来。双向信道电缆描述符允许有多于一个物理信道（即一个不同的回传 RF 载体）可用。

应用会等待主机用指令 Connect _ Ack 应答。

连接结果

如果调制解调器不支持此连接标识符，也就是说，如果资源是一个串行端口，主机应该返回一个返回值（return _ value）为“不支持的连接”（CONNECTION _ NOT _ SUPPORTED）的 Connect _ Ack 指令。如果主机不能通过电话或电缆信道建立连接，它会返回一个返回值（return _ value）为“失败（FAIL）”的“Connect _ Ack”指令。当调制解调器没有预定的传输时，应用应该离开这一会话。为了警告用户连接问题，也许可以建立一个 MMI 会话。

当连接到调制解调器后，主机应该返回一个返回值（return _ value）为“确定（OK）”的 Connect _ Ack 指令。然后应用可以为通信会话设置参数。

G7.9.6 设定参数

指令“设置参数（Set _ Params）”允许设置缓冲区大小和超时参数。参数缓冲区大小（buffer _ size）指主机需要存储的接收到信息的长度。参数超时指为了查找收到信息的结束必须等的最大时间。当应用收到主机的 Set _ Params _ Ack 指令时，它就可以开始进行数据交换了。

G7.9.7 数据交换

G7.9.7.1 接收数据

当资源收到数据，它首先开始填充它的缓冲区0。然后它探测信息的结束：超时或缓冲区已满，它开始用下一个收到的数据填充缓冲区1。

应用可以通过指令Get_Next_Buffer请求接收的缓冲区，它首先使用comms_phase_id“0”，然后交替的1，0，1，0，主机会根据次序正确与否返回一个返回值为“确定（OK）”或者“错误（ERROR）”的Get_Next_Buffer_Ack指令。

当主机收到一个完整的缓冲区，而且应用已经请求相应的部分，它就会通过comms_rcv指令返回信息，首先是comms_phase_id“0”，然后交替地1，0，1，0，如果应用没有请求一个缓冲区，主机就不能发送它，而且直到主机返回了相应的缓冲区，应用才可以请求它（除了第一次），如果应用试图这样做，那么主机将返回返回值为“错误”的Get_Next_buffet_Ack指令。

G7.9.7.2 发送数据

当应用想发送数据时，它使用comms_send指令，首先是comms_phase_id“0”，然后交替的1，0，1，0，主机将返回Send_Ack指令，返回值“确定”说明信息已经发送，如果次序不正确，则返回值为“错误”。只要主机没有应答前一个，应用不能发送缓冲区（除非是第一次），如果应用试图这样做，那么主机会返回返回值为“错误”的Send_Ack。

G7.9.8 断开信道

当应用想终止与调制解调器的传输时，它发送一个“断开信道（Disconnect_on_Channel）”的命令。

如果资源不支持连接的话，例如一个串行端口，主机会返回一个返回值为“不支持的连接（CONNECTION_NOT_SUPPORTED）”的Disconnect_Ack指令。

当调制解调器断开后，主机用返回值为“确定（OK）”的Disconnect_Ack回应。

G7.9.9 关闭会话

当应用不再需要调制解调器时，它应该关闭与它的会话。

G7.9.10 低速通信传输的例子

见表G4。

表G4 低速通信传输的例子

应用	指令	主机
请求调制解调器类型资源的会话	(...open session...) <——>	如果有闲置资源，允许会话
请求连接信道	comms_cmd (Connect_on_Channel) ——>	
	Comms_reply (Connect_Ack) <——	成功连接调制解调器

表 G4 低速通信传输的例子（完）

应用	指令	主机
给主机发送通信参数	Comms _ cmd (Set _ Params) ——>	
	Comms _ reply (Set _ Params _ Ack) <——	设置通信参数
发送信息：用 comms _ send _ more/last 命令	comms _ send (phase 0) ——>	
长度长于缓冲区大小	comms _ send (phase 1) ——>	
	Comms _ reply (Send _ Buffer _ Ack，phase 0) <——	当第一部分数据发送后，an ack. 被返回了
	Comms _ reply (Send _ Buffer _ Ack，phase 1) <——	当最后部分数据发送后，an ack. 也被返回了
需要接收一个信息	Comms _ cmd (Get _ Next _ Buffer，phase 0) ——>	
	comms _ reply (Get _ Next _ Buffer _ Ack，phase 0) <——	请求已经答复
	comms _ rcv (phase 0) <——	缓冲区 0 已经收到了一个信息：用 commands comms _ rcv _ more/last 指令
需要接收的下一部分信息	Comms _ cmd (Get _ Next _ Buffer，phase 1) ——>	
	comms _ reply (Get _ Next _ Buffer _ Ack，phase 1) <——	返回 an ack.
	comms _ rcv (phase 1) <——	缓冲区 1 收到了信息
数据交换终止，需要断开调制解调器	comms _ cmd (Disconnect _ on _ Channel) ——>	
	Comms _ reply (Disconnect _ Ack) <——	调制解调器断开
会话关闭	(...close session...) <——>	资源释放

G7.9.11 强制要求

G7.9.11.1 对于电话调制解调器类型资源

此资源支持所有指令。连接描述符类型（connection _descriptor _type）的值是01。如果收到其他的值，主机会返回错误“不支持的连接”。

G7.9.11.2 对于双向电缆信道资源

此资源支持所有指令。连接描述符类型（connection _descriptor _type）的值是02。如果收到其他的值，主机会返回错误“不支持的连接”。

G7.9.11.3 对于串行端口资源

此资源支持所有指令。如果资源收到“连接到信道（Connect _on _Channel）”或者“断开信道（Disconnect _on _Channel），主机会返回错误“不支持的连接”。

G7.9.12 推荐参数设定

不同的资源类型代表低速通信资源的多重实例，而且多个资源类型代表的一个实例也许表示有不同的速度选项可用。然而，每一资源实例只支持一个会话。

G7.10 确认

这一可选的资源的定义是为了方便主机到模块的确认，特定的主机/模块组合也许会需要它。在特定的情况下，当不指望模块对确认的支持时，主机必须允许它和模块之间的未确认的相互操作。如果这一模块声称它自己是支持已定义的确认过程类型的那种类型，这种确认过程只能用来拒绝一个模块的服务。

推荐参数设定

确认资源应该至少支持一个会话。

G7.11 图文电视广播显示

提供这一可选资源是用来允许应用访问主机提供的图文电视广播显示系统。它是MMI资源的一个可选的屏幕显示机制，而且可能会适用于某些应用。

G7.12 智能读卡机

这一可选资源允许应用访问主机上的或者另一个模块上的智能读卡机。这不是通常的条件接收卡使用的智能读卡机，如果在系统中使用它，它会是一个对其他基于卡的相互操作的可用的读卡机，像银行卡或电子钞票卡操作。

推荐参数设定

协议允许在任何时间有多于一个的活动的会话。然而，在任何时候，这些会话中，只有一个能“连接”到读卡机。没有连接的会话只能读取读卡机和卡的状态信息。智能读卡机资源的每一个读卡机资源实例应该支持高达16个会话。

G7.13 EPG未来事件支持

提供这一可选资源可以使电子节目指南从一个CA应用得到未来事件的字幕状态信息，当然，需要有这种可用信息。这一协议也包括一种设备，它允许条件接收MMI会话发生并且如果要求的话，给出字幕，就像按次计费事件的购买一样。注意此资源是由EPG提供的，而且是否创建一个会话来提供所需的信息是由CA应用决定。

推荐参数设定

EPG未来事件支持资源应该至少支持16个会话。

G8 出错管理

G8.1 概述

附录F未定义应用层出错信息，而是假定主机和模块卡都符合规范。但设计可靠的软件在因规范实现中存有缺陷而引起出错的情况下，不应死机。

本附录旨在定义一些实现时的规则，以改善附录F的不同实现之间的互操作性。

本附录不包括与低层相关的出错信息，而是假定与服务资源的会话建立时会话层向应用层传送的应用协议数据单元（APDU）是正确的。

G8.2 应用层失败

G8.2.1 协议错误

协议错误可能是因进程处于不合适状态而致使无法处理命令引起，比如：应用程序请求调制解调器发送数据，但此前尚未设置通信参数或未建立与服务器的连接。

G8.2.2 命令错误

命令错误可能是因收到错误的应用协议数据单元APDU，如不支持的应用对象标记、不支持的数据长度、无法理解的数据域。

G8.2.3 传输层断开

传输层断开（由于物理层拔开模块卡或由于逻辑上将传输层连接删除引起）可引起应用层出错。比如，当使用某服务资源的应用程序断开时应释放该资源，否则该服务资源将永远无法使用。

G8.3 实现指南

G8.3.1 传输层断开错误

传输层断开将影响会话层、资源管理器、应用层信息和提供CA支持的各种实现。

如果该传输层连接是为某个应用程序创建，则所有正被该应用程序使用的资源都应释放。

—主机会话层的处理规则

主机将给所有远端服务资源发送关闭会话请求（Close_Session_Request），服务资源提供者（主机或模块）负责恰当地释放资源，如断开低速通信或关闭人机界面（MMI）对话；

—应用程序信息和提供CA支持的服务资源的处理规则

它们应从自己的应用程序列表中删除已断开的应用。

如果断开的传输层连接是为某服务资源提供者创建的，则所有打开的会话应关闭；

—主机会话层的处理原则

主机将给所有应用程序发关闭会话请求（Close_Session_Request），而忽略任何使用已断开的服务资源的请求；

—资源管理器的处理原则

资源管理器应更新其资源列表。若因此产生变化应给所用应用程序发送设置更改消息（Profile_Change）。

G8.3.2 命令错误

—通用规则

若服务资源或应用程序收到的命令中应用对象标记（TAG）未指定或与自己无关，它应忽略该命

令，不做任何处理。

G8.3.2.1 资源管理器

—设置查询

忽略任何长度非零的命令；

—设置更改；

—忽略任何长度非零的命令；

—设置应答

长度和数据取决于资源管理器支持的服务资源数目。合理的支持资源数目能达到 64 个。主机至少要提供以下服务资源：资源管理器、应用程序信息、CA 支持、主机控制、日期时间、人机界面和低速通信；

—应用程序层规则

任何长度参数不等于 4×N 的命令都是错误的并被忽略，其中 N 的取值范围为 7 到 64。

忽略任何应用程序不支持的服务资源标识符；

—资源管理器规则

任何长度参数不等于 4×N 的命令都是错误的并被忽略，N 的取值范围为 0 到 57。忽略任何含有不被支持的服务资源标识符 resource _ indentifier () 的命令，即任何取自资源管理器、应用程序信息、CA 支持、主机控制和 MMI 中的服务资源标志符只能由主机提供。任何命令，如其列表中包含重复的服务资源标识符，都被忽略。服务资源标识符应是唯一的。

G8.3.2.2 应用程序信息

应用程序信息查询

忽略任何长度非零的命令。

应用程序信息

菜单字符串长（menu _ string _ length）指示菜单字符串中字符个数，最长为 40，故长度参数最大值为 46。允许菜单字符串长度为零，此时主机将会实现缺省菜单字符串；

忽略任何长度参数超出 6 到 46 范围的命令；

忽略任何未指定应用类型（application _ type）的命令；

忽略任何包含未指定文本字符（text _ char）的命令。

进入菜单

忽略任何长度参数非零的命令。

G8.3.2.3 CA 支持

CA 信息查询

忽略任何长度参数非零的命令。

CA 信息

CA 应用可支持至少 1 种至多 16 种 CA 系统标识号（CA _ system _ id）看上去是合理的；

主机应支持任何种类 CA _ system _ id 但不应解读该 CA；

忽略任何长度参数超出 2 到 32（含）范围的命令。

CA 节目映射表

GB/T 17975.1 规定传送流节目映射分段（TS _program _map _section）不超过 1024 字节。该限定对 CA 节目映射表也是合理的因为只需要传输 CA _descriptor；

忽略任何长度参数超过 1 018 字节的命令；

因为 CA _descriptor 最长为 256 字节，故忽略节目信息长度 program _Info _length 或原始流信息长度 ES _info _legnth 超过 257 的命令；

忽略任何未指定 CA _pmt _list _management 的命令；

忽略任何未指定 CA _pmt _cmd _id 的命令；

忽略含有节目层或 ES 层均不支持的 CA 描述的命令，如含有不支持的 CA _system _id；

忽略含有不存在的或无关的基本流 PID 的任何命令；

所有预留域应置"0"。

CA 节目映射表应答

忽略含无关节目号（program _number）的命令；

忽略含无关的版本号/当前_下一个指示（version _number/current _next _indicator）的命令；

忽略含不相关 CA _enable 的命令；

忽略含有不存在或不相关基本流 PID 命令；

所有预留域应置"0"。

G8.3.2.4 主机控制

调谐器

忽略含长度参数不等于 8 的命令；

忽略含有未指定的网络标识号的命令；

忽略含有未指定的原始网络标识号命令；

忽略含未指定的传送流标识号命令；

忽略含有未指定的业务标识号的命令。

替换

忽略长度参数不等于 5 的命令；

忽略含有未指定或不相关的（replaced _pid）（如预留 PID）命令；

忽略含有未指定的 replacement _PID 或不相关的（如预留 PID）或不兼容的（如视频替换音频）命令；

每个应用中最多替换索引数目为 256（0 到 255），同一替换索引中最大被替换的最大 PID 数目是有限的，比较实际的最大数目为 16。

不加扰透明替换

忽略长度参数不等于 1 的命令；

忽略含有未指定或不允许的被替换索引（replace _ref）的命令。

请求释放

忽略长度参数非空的命令。

G8.3.2.5 日期和时间

日期时间查询

忽略长度参数不等于 1 的命令。

日期时间

忽略长度参数不等于 5 或 7 的命令；

忽略含有未指定的国际时间编码（UTC _ time）的命令。

G8.3.2.6 人机界面（MMI）

关闭人机界面

忽略长度参数不等于 2 的命令；

忽略含未指定 Close _ mmi _ id 的命令。

显示控制

忽略长度参数不等于 1 或 2 的命令；

这条命令对一般规则是个例外：会在显示应答命令中传送回应或确认信息；

对于含有未指定的 display _ control _ cmd 命令，将回应 display _ reply，其中 display _ reply _ id 等于 unknown _ display _ control _ cmd；

对含有未指定 MMI _ mode 的命令将回应 display _ reply，其中 display _ reply _ id 等于 unknown _ mmi _ mode。

显示回应

忽略含有未指定的 display _ reply _ id 的命令。

按键控制

忽略长度参数不在 1 到 19 范围的命令；

忽略含未指定的 Keypad _ control _ cmd 的命令；

忽略含有未指定的 Key _ code 的命令。

按键

忽略长度不等于 1 的命令；

忽略含未指定 Key _ code 的命令。

文字

忽略长度参数超过 40 的命令；

忽略含未指定 Text _ Char 的命令。

查询

忽略长度参数超过 2 到 42 范围的命令忽略含未指定 Text _ Char 的命令。

应答

忽略长度参数超过 1 到 41 范围的命令；

忽略含有未指定 answ _ id 的命令；

忽略含未指定 Text _ Char 的命令。

菜单

忽略含有错误 TEXT（）的命令；

忽略 choice _ nb 不等于 255 且超出 0—10 范围的命令；

忽略包含超过 25 组命令 TEXT（）的命令。

菜单回应

忽略长度参数不等于 1 的命令；

忽略含无关 Choice _ ref 的命令。

列表

忽略含有错误 TEXT () 的命令；

忽略 choice _ nb 不等于 255 且超出 0—10 范围的命令；

忽略包含 25 组命令 Text () 的命令。

G8.3.2.7 低速通信

通信命令

忽略含未指定 comms _ command _ id 的命令；

忽略含错误连接描述子 (connection _ descriptor) 命令；

忽略缓冲区大小参数超过 254 的命令；

忽略含 comms _ phase _ id 不等于 0 或 1 的命令。

连接描述子 (connection _ descriptor)

含未指定连接描述子类型连接描述子将出错；

含有不支持的电话描述子的连接描述子将出错；

含有不支持的通道标识号 (channel _ id) 的连接描述子将出错。

通信应答

忽略长度参数不等于 2 的命令；

忽略含未规定 comms _ reply _ id 的命令；

忽略未规定 return _ value 或 return _ value 与 comms _ reply _ id 无关的命令。

通信发送

忽略长度参数超过 255 的命令；

忽略未规定 comms _ phase _ id 的命令。

通信接收

忽略长度参数超过 255 的命令；

忽略未规定 comms _ phase _ id 的命令。

G8.3.3 出错时的行为

G8.3.3.1 服务资源的行为

一般规则

忽略任何未预期的命令 (当协议出错时)，然而有些情形下如有死锁危险，太多错误或超时错误，资源可决定关闭与出错应用程序的会话。

应用程序信息服务资源的行为规则

适用一般规则。主机可删除出错的 CA 应用程序并给用户报错。

提供 CA 支持资源的出错处理规则

适用一般规则。主机可删除出错的 CA 应用程序并报错给用户，如正处理人机对话界面，主机可关闭该对话。

G8.3.3.2 应用程序的行为规则

一般规则

忽略任何不被期待的命令（当协议出错时），然而有些情形下如有死锁危险，太多错误或超时错误，应用程序可决定关闭与出错资源的对话。

G9 通用实现指南

G9.1 初始化

考虑两种情形：第一种情形将模块卡插入已加电开机的主机中；第二种情形主机包含两个或多个模块时开机。

当模块插入时，先接通电源和地脚，然后是数据和控制信号脚，最后是模块检测脚。模块执行上电复位和其他一些操作，将输出信号脚置成高阻，因此没有信号会干扰已存在模块的操作。当主机检测到模块插入时，它从模块的属性存储器中读取卡信息结构体，从中判断出操作电压（与电压感应脚状态一道）和是否是数字电视广播公共接口模块，若不是则初始化过程就此停止。若是数字电视广播公共接口模块，则主机中传送流旁路接口被关断，并将适当值写入模块卡配置寄存器中，使模块处于工作状态。此时 MPEG-2 传送流将送给模块卡，但还未被解扰。由于模块卡附加进来了延迟，此处不可避免地会在传送流中引入几字节错误数据，PC 卡初始化过程到此完成，详细细节在 PC 卡标准中给出。在这一过程中，模块已执行上电复位，等待 PC 卡配置结束，同时还进行其他一些配置以使信号进入工作状态。

主机通过执行缓冲区大小协商进程，初始化物理层命令接口。对链路层不需要显式的初始化。主机创建与模块卡的传输层连接。一旦连接建立，模块请求打开与主机资源管理器的会话，资源管理器拥有早就已知的资源标识号，且一旦能保证会话建立成功，资源管理器开始向模块发配置查询命令，以查清模块卡是否提供服务资源。模块发回配置应答，若提供服务资源应答中将含服务资源列表，否则列表为空，接着资源管理器向模块发配置改变命令，大多数情况下模块回应配置查询命令，主机回应配置应答命令，列出所有可供模块使用的服务资源。若模块只是单纯提供服务资源，则可不执行上述最后一步，因为它不需使用主机提供的资源（除了资源管理器外）。如果在前一步骤中模块已回应了它所提供的服务资源列表，则主机将更新其主服务资源列表，若此列表发生变化，主机也向所有其他活动的传输层连接发送配置更改对象，至此模块卡就绪，可执行其任务。

应用模块将创建与应用程序信息服务资源的会话以传送应用程序信息，并管理应用程序菜单入口。一旦创建了会话，主机就向应用程序发应用信息查询命令，应用程序向应用程序信息回应。

条件接收应用也创建到 CA 支持服务资源的会话，以允许应用程序获取 SI 中的 CA 信息和关于用户所选业务的信息。一旦会话建立，主机向应用程序发 CA 信息查询，应用程序回应 CA 信息，主机接着可进行与 CA 应用程序的一系列对话过程以控制那些选定的业务可被 CA 应用解扰和在何种条件下解扰。

至此初始化过程结束。

另一种情形是当主机包含两个模块卡开机时，过程相当简单。主机执行一遍自己的初始化进程，同时模块执行上电复位并等待。主机将顺序或并行执行 PC 卡初始化步骤，取决于主机实现。前面所述的上层初始化也可顺序或并行执行。然而一种有用的优化是对所有模块一起执行服务资源管理器协议。这种情况下，主机对所有打开的服务资源管理器会话执行配置查询命令，以便在发出任何配置更改命令对象前收集所有可能的服务资源信息，这可通过有意使用超时和主机中关于当前模块（和传输层连

接）的数目的上下文信息来实现。

G9.2 断开连接

当模块卡拔出时，模块传感检测引脚首先断开，故而在数据和电源引脚断开前还有一小段时间便于主机自我清理，主要完成的功能是在主机上为 MPEG-2 传送流重开主机旁路接口。这将会因为模块引起的延迟突然消失而引入对数据流的干扰，但无法避免。

G9.3 热插拔

设计时必须考虑用户会在主机工作过程中随时插拔模块卡，因此主机和模块都必须设计成支持这种操作。在 PC 机中一般这样处理：卡未插入时，通过接口器件隔离 PC 卡槽，只有检测到卡插入才加电，而为设计低成本主机，将公共接口的命令接口部分当作总线来操作更为便利。这种情况下 PC 卡槽上的一些数据和控制脚一直保持激活状态，故主机和模块设计必须能尽量减少插入模块时总线上的误操作。

PC 卡槽上不同信号引脚设计成不同长度。电源和地引脚最长（5.00 mm）信号线次之（4.25 mm），卡检测脚最短（3.50 mm）。假定以合理的速度插卡，电源脚连接后 3 ms～5 ms 是信号脚接通，再过 3 ms～5 ms 才是卡检测脚接通。

卡插入时有两种原因可能导致误操作。第一种是因为模块卡上电源线电阻很小，会引起电源接收端电压短时降低。通过限制模块卡中电源线电容，主机电源引脚采用合理的储能电容以提供浪涌电流，并在主机储能电容和模块卡电路间提供一较小但不可忽略的阻抗（R 和/或 L）。C，L，R 的取值须设计适当，以使模块电源上升时间在 3 ms 内，同时模块卡设计应使所有信号在 3 ms 内处于高阻状态。

第二种可能是插卡时引起信号脚功能异常。这不应是由电阻性负载引起而是由于模块中电容性负载引起—主机中信号线将向模块节点电容提供充电电流脉冲。基本方法是尽量增大充电时间，因而可尽量减少充电电流峰值。这需要一些设计技巧如尽量降低主机中总线电阻，但适当增加引向各接口槽信号引脚的电阻。通常主总线使用宽 PCB 布线而分支信号线使用较细 PCB 布线来实现。

这样做增加了串联的 L 和 R 值，一旦模块卡上电，信号线将处于高阻状态直到主机检测到卡插入并开始执行读/写卡操作。在拔卡时主机有 3ms 以上时间清除与模块卡的连接。这涉及关闭会话和传输层连接。模块卡较少会检测拔卡操作，但使用卡检测引脚且不违反关于这些信号引脚的操作规范也是可能的。

G9.4 协议层实现

在设计象本文这种分层规范时，使用一种可应用于所有分层或大部分分层的基本设计思路是有益的。随着数据包在主机和模块间交互流动，在各分层间会传送许多数据和控制交换信息，对此使用一种规范的模板会使各层的编写更为容易，因为在编写某层所用的许多设计方法也可以用于其他层。下列讨论建议了一两种可能的方法但决非全部，以前用其他方法做过此类设计的软件设计者可能希望采用自己的经验方法。

某层和下一层的交互功能大概可分解为 4 种基本操作：向下层发送、从下层接收、对下层做某项操作、被下层告知发生某个事件。这样就允许在上下层间交换数据，创建或删除连接，并在连接发生或消除时产生通告。对上下层间交互作这些限定后，也使设计中带进到分层间未预测的和不希望出现的交互作用的可能性大为减少。

层间通信可简单归结为函数调用，显然意味着一大串函数调用在分层之上或之下建立起来，除非用于堆栈的内存太小，否则这不会成为问题，但适当安排所有这些调用的顺序变得重要起来。比如，若

创建了到服务资源的会话连接后，则回应将送回到创建该会话的应用程序，重要的是保证对该服务资源的“创建”会话操作不会导致对该资源启动应用层协议，而只能在回应送回之后才启动。适当的排序是为该会话创建供会话层使用的管理数据，然后将应答返回给请求的应用程序，此后才通知创建到服务资源的会话。

另一种层间通信的方法是在分层间使用数据和命令队列（可以是共享队列）。这种情形下容易保证释放控制之前在分层内完成所有需要的操作，然而这种实现方式更为复杂，因为所有控制都必须返回给某种排序器，由排序器依据各个层间队列中的内容决定将控制交给哪一层。在应用层自身，规范各服务资源和会话层间的接口也是重要的。通过会话层使用的管理数据结构与各资源的接收函数结合起来，比如使用函数指针，可高效的实现消息发送机制，而不必使用 Switch 或 case 语句或更糟情形使用一连串 if 语句。

G10 使用环境

G10.1 机械尺寸指南

标准要求主机接受 I 型和 II 型 PC 卡，对 III 型卡支持是可选的，除上述要求外，主机设计时应在主机外预留出标准 PC 卡尺寸规格的扩展空间。这允许 PC 卡本身扩展尺寸超出标准卡大小，也为从卡上引出电线创造条件。

扩展大小限定如下，并参考图 G3：

厚度不应超出 PC 卡的尺寸参数 T；

可在 PC 卡（PC 卡加长部分）每边加宽最多 2.1 mm；

可对 PC 卡加长 20 mm；

应能允许一根很长线缆从卡上引出。

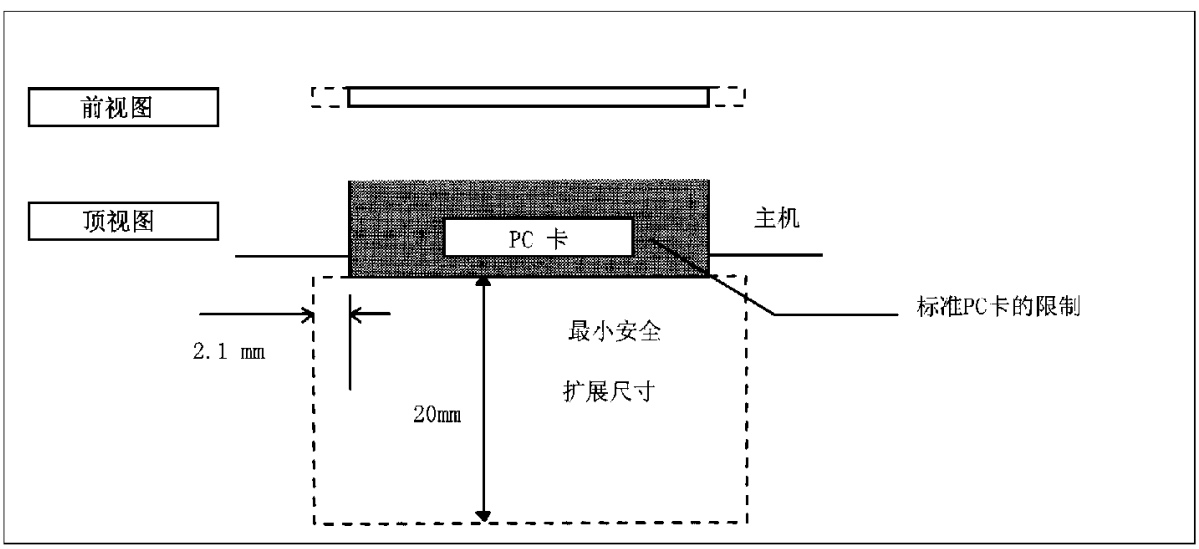


图 G3 PC 卡扩展大小

G10.2 模块卡使用环境考虑

附录 F 要求可插拔模块在产品设计寿命周期内可工作于高达 55℃的环境温度而可靠性不会有任

何降低。规范限定当模块满净荷运行散热时主机内模块周围最多升温 15℃，在此假定主机和模块都已上电工作足够长时间，工作温度已经稳定，这一限制对保证可插拔模块中可能使用的锂电池的最高工作温度范围是必要的。通常限制在 55℃～60℃之间。

规范承认有少数主机在经常高于 40℃的环境下工作可能会超过 55℃的模块环境温度，这些模块的电池寿命会缩短，具体取决于有多长时间超出了电池厂家规定的温度极限。电池寿命减少的程度由电池生产商给出的超常规工作曲线给出，然而消费电子设备的使用环境变化多样，难以准确预计模块温度会超出 55℃常规极限值的时长。

主机和模块厂商有责任采取合理而适当的措施以保证模块工作环境的适当温度。

G10.3 主机设备生产厂商指南

主机厂商必须考虑装有电池的模块将会在主机中使用，必须采取措施保证工作环境符合规范。这意味着为模块插槽选定较好位置以便将模块环境温度保持在极限内而不致影响电池寿命。这包括热力学分析以判定最佳模块位置和空气流通测量及模块卡以最大功率工作时对设计的验证测试，还需考虑的问题包括模块朝向和多模块主机中模块间离的多近。总体而言，使用垂直朝向更易散热，模块间须充分分开以防热量聚积在模块中间。

G10.4 模块卡厂商指南

模块厂商必须选择适当的电池供电技术以便最好地承受住可能会遇到的高温工作环境。超常规工作曲线最适合以上温度极限，应作为选择电池的部分依据。厂家应将电池放在 PCMCIA 卡物理规范第三卷图 3 推荐的位置，也应将发热器件置于卡中离电池位置尽量远的地方。

模块厂商应意识到主机厂家可能为 PC 卡上推荐的电池位置提供了特制的空气流通设计，一般位于离卡连接插口对端约 2.0 cm 区域内。

模块厂家应将使用电池的持续能耗降到最小以达到模块工作所要求的必需能耗。要优先考虑只在模块拔下时才消耗电池电能的设计，因为这样电池寿命可能接近闲置时的寿命，尽管有可能不时地置于高温环境下。

G10.5 模块接地

G10.5.1 设计原理

电磁兼容（EMC）是个特别难以处理的领域，特别是在象公共接口这样的情形中，难以划分模块和主机对电磁兼容性问题的责任。对此问题没有简单的解决方案，从现存市场（如 PC 卡）没有可用于这种划分的裁决步骤也印证了这一点。这里采用的方法是强烈建议采用接地金属回路网（CLIPS）并推荐一些合理的工程实践做法。

G10.5.2 指南

下列设计原则选自良好的工程实践做法。只有对最后实现的 EMC 性能有充分把握时下列给出的要求细节才可放宽。

在主机上要用双接地金属网条，为增加电磁干扰屏蔽效果，可铺设一层在离模块 5mm 内与模块平行的接地层，该接地层直接连到接地金属网条上。类似地若有金属底盘接地金属网条也直接连到该金属盘上，模块可经此底盘伸出来，该金属底盘可成为封闭电磁屏蔽罩的一部分。选择接地金属网条形状时要考虑到模块中可能含有智能卡。

模块插座上 4 根接地引脚到主机电气地的引线电阻应尽量小，一种做法时将接地脚直接焊到 PCB

接地层上（接地脚引线最长 10 mm）用 PCB 布线传与模块连接的信号。各种情形下都要保证主机输出到模块的信号和本规范中模块输出到主机的信号遇到阻抗时（包括可能存在于输入脚和电源脚上的馈入信号和反射信号）发出的电磁干扰不会超标，建议布线时做传输线和天线理论分析以预测将产生的电磁干扰水平。

在模块上要按 PC 卡标准准备两处接地金属点，除非已按下段描述准备好了。每个接地点被连到模块的金属盖（可能两块金属盖隔离每接地点连到一块上，也有可能只有一块金属盖连到两个接地点上）。金属化外形指外包导电金属塑料或覆盖模块的整个表面的模压成型金属盖。大多数情况下，防止从接地金属网条到模块信号地之间出现一条小电阻通路以防止接地回路和天线效应是有益的。4 根信号地引脚应经用极小电阻通路引到主机电气地上，一种做法是用最短布线就近引到覆盖 PCB 大部分区域的接地层上。

带智能卡的模块卡在整个 3.3 mm 条深的导槽区不必提供接地金属网条。在导槽的两边可留 1.1 mm区域不接触以便插入智能卡，设计主机接地金属网条时要考虑这个问题，还要考虑引导槽中可能为空（智能卡已拔出）

关于静电保护方面，模块须能承受插拔时电源脚的放电和外壳到内部电路板上的放电，主机必须能经受到电源脚的放电。

G11 限定和说明

比特和字节顺序说明

传送流接口和命令接口都是面向字节的，字节中最高有效位定义为 D7，因而字节中比特顺序也就定了。字节顺序定义如下：对多字节参量，先传高有效字节。这就是所谓 bigger-endian，接口硬件工作方式都统一，因此一串流的到达字节被依序放入内存中，由低到高地址区域，故一个数的最高有效字节位于该数所占内存的最低地址字节。依这种方式操作的处理器也叫 bigedian，用这些处理器不需要在软件中作顺序转换。然而有些处理器系列采用 little-endian 方式，此时一个数的最高有效字节位于该数所占内存区的最高地址区，用此处理器执行的软件需要重新排序字节才能正确操作。

注意：当配置 PCMCIA 接口时使用 little-endian 方式读取卡信息结构（card info. Struct）也就是说，多字节数高有效字节存于 PCMCIA 卡属性存储区的高位地址区，需注意 PCMCIA 卡数据和其上各层数据格式上的这种变化。

在实际中，处理器还有一公共特性即地址对齐方面的限制，那意味着上述两大类处理器可能都需要组合了提取，存放和转换功能的子函数。例如，当分析某收到的消息时，消息中含有 32 比特无符号高有效位在前整型数，该数在消息缓冲区中的地址可能会使直接以 32 比特访问的处理器产生总线地址对齐错误，一个合用的提取子函数须逐字节地读出该数，以正确顺序放入地址已适当对齐变量中，以便后续应用程序使用，类似地一个合用的存放子函数须在组装消息时以正确顺序将多字节数放入缓冲区任意地址。这类子函数对所有软件实现都是不可缺少的。

G12 CA _ PMT 操作的示意图和流程图

G12.1 描述 ca _ pmt _ list _ management 参数使用的示意图

下列图表给出使用 ca _ pmt _ list _ management 参数的一些例子。每个箭头对应于发送一个 CA _

PMT 参数。CW table 界线指出 CA 应用程序存储哪个 CW（例如：cw1 意味着 CA 应用程序把节目 1 的控制字存入存储器）。

—选定 1 个节目的 CA _ PMT 管理的示例

在这个示例中，用户一次仅选择一个节目，用户从一个节目切换到另外一个。如图 G4 所示。

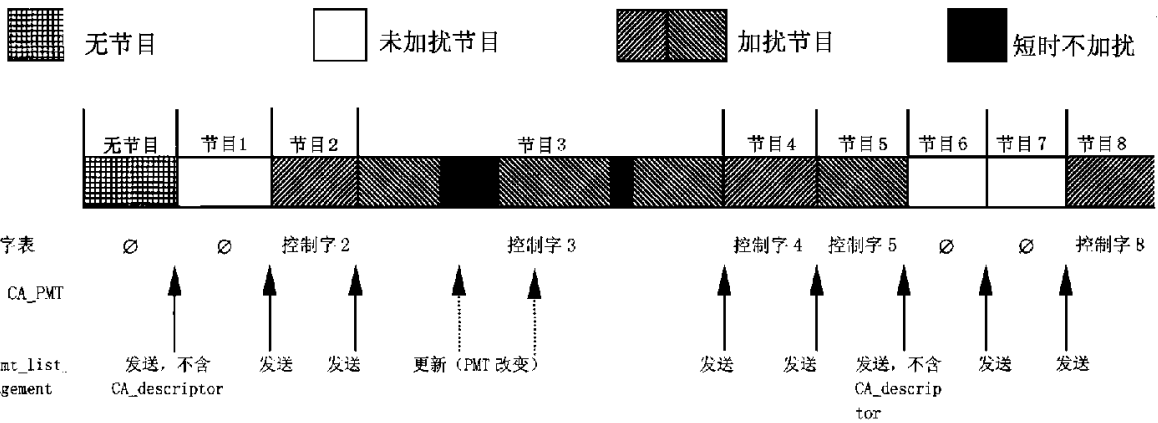


图 G4 选定 1 个节目的 CA _ PMT 管理的示例

—选定 3 个节目的 CA _ PMT 管理示例

这个例子中，用户并行地选择了几个节目（多达 3 个）然后又取消选择。最后，没有选定任何节目时，主机发送一个 CLOSE _ SESSION 对象关闭与 CA 应用资源的会话。如图 G5 所示。

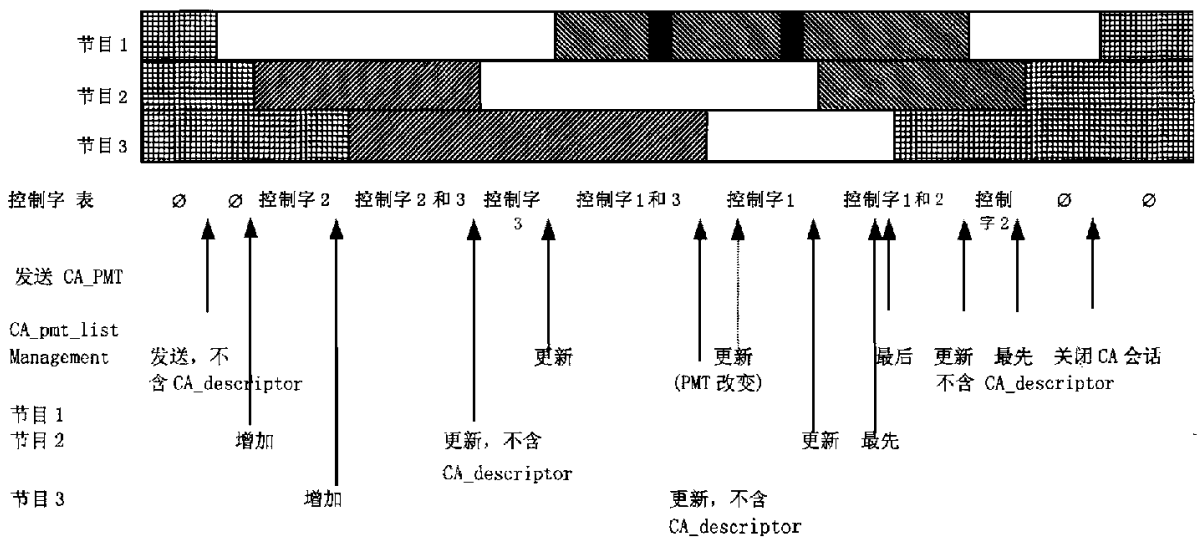


图 G5 选定 3 个节目的 CA _ PMT 管理示例

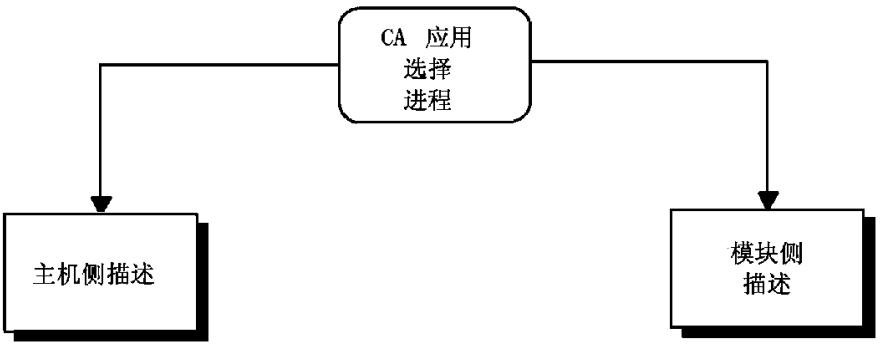
G12.2 描述 CA _ PMT 和 CA _ PMT _ reply 使用的流程图

下列流程图描述以下几种情形下主机和模块如何使用 CA _ PMT 和 CA _ PMT _ reply：

a）选择唯一节目（未描述 ca _ pmt _ list _ management）；

b）同样的 ca _ enable 应用于选定节目的所有基本流（在几个 CA 模块间分离解扰的情形不予描述，当考虑下列流程图中描述的一个完整节目的CA应用。选择节目可由模块分别对该节目各基本流来

施行时，容易由这些流程图推导出分离解扰时的流程图）。见图 G6 到 G12。



重要说明：下列流程图所描述主机进程可在任何时刻被一外部事件所打断（例如：用户选择另外一个节目）

重要说明：下列流程图中所描述的模块进程可在任何时刻被一外部事件打断（例如：用户选择另外一个节目）

图 G6　CA _ PMT 和 CA _ PMT _ reply 使用的流程图之一

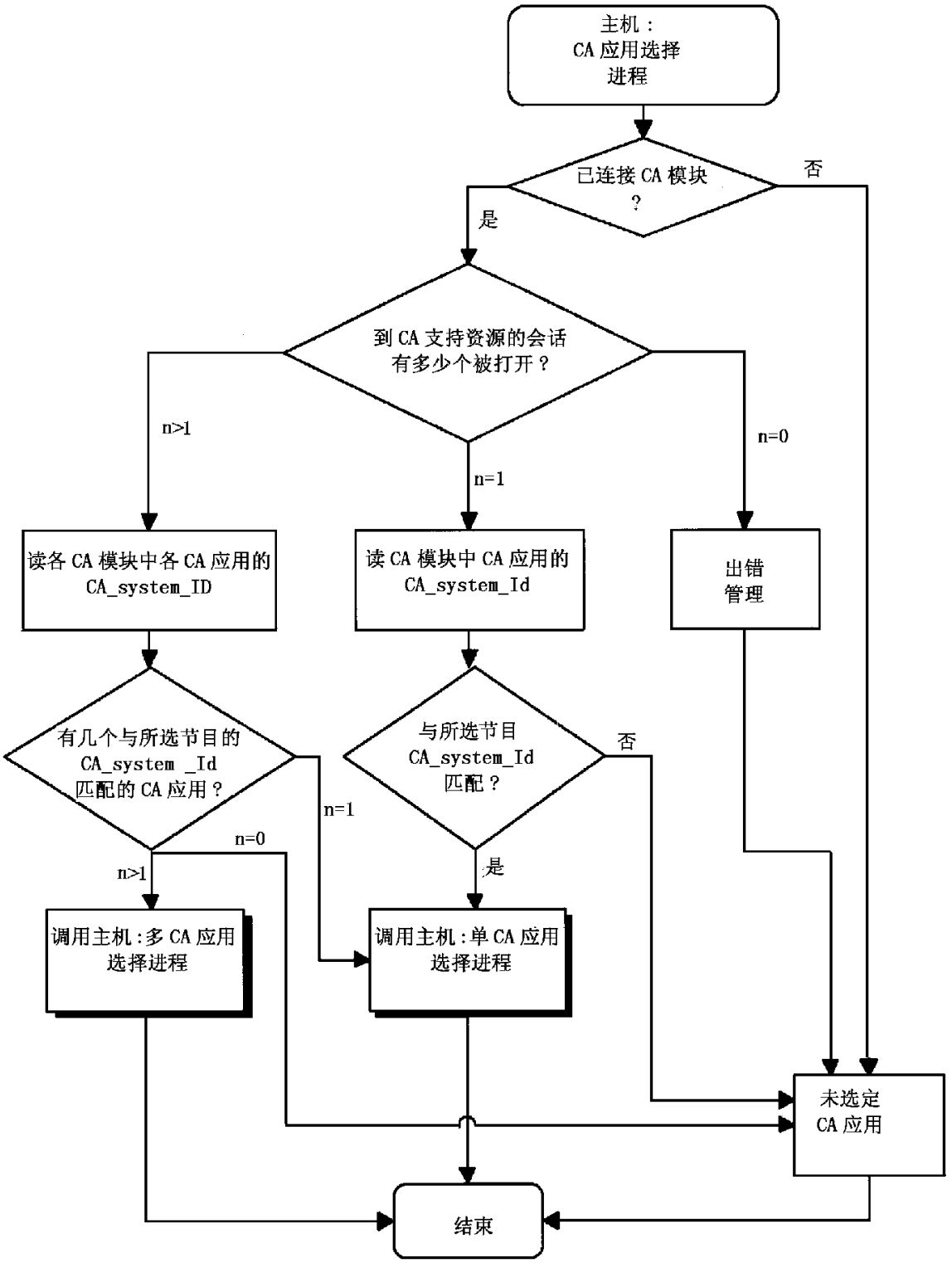


图 G7　CA _PMT **和** CA _PMT _reply **使用的流程图之二**

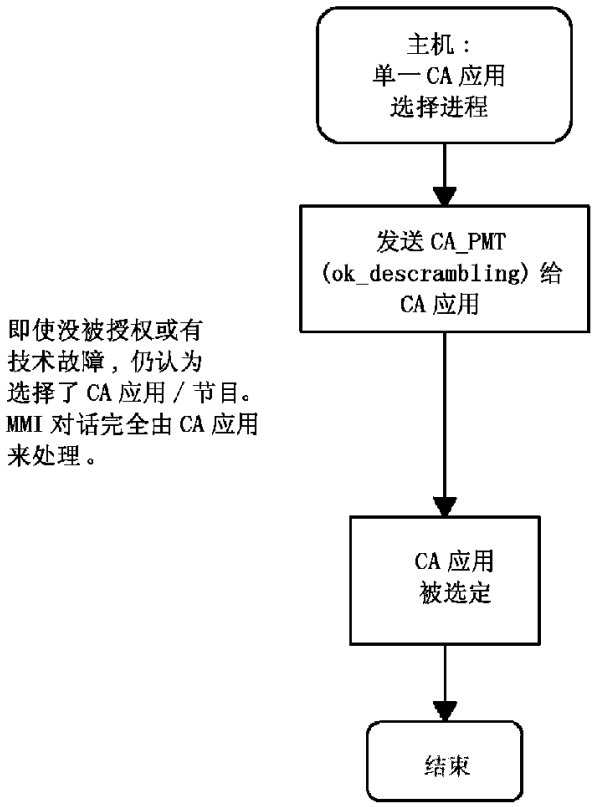


图 G8　CA _ PMT **和** CA _ PMT _ reply **使用的流程图之三**

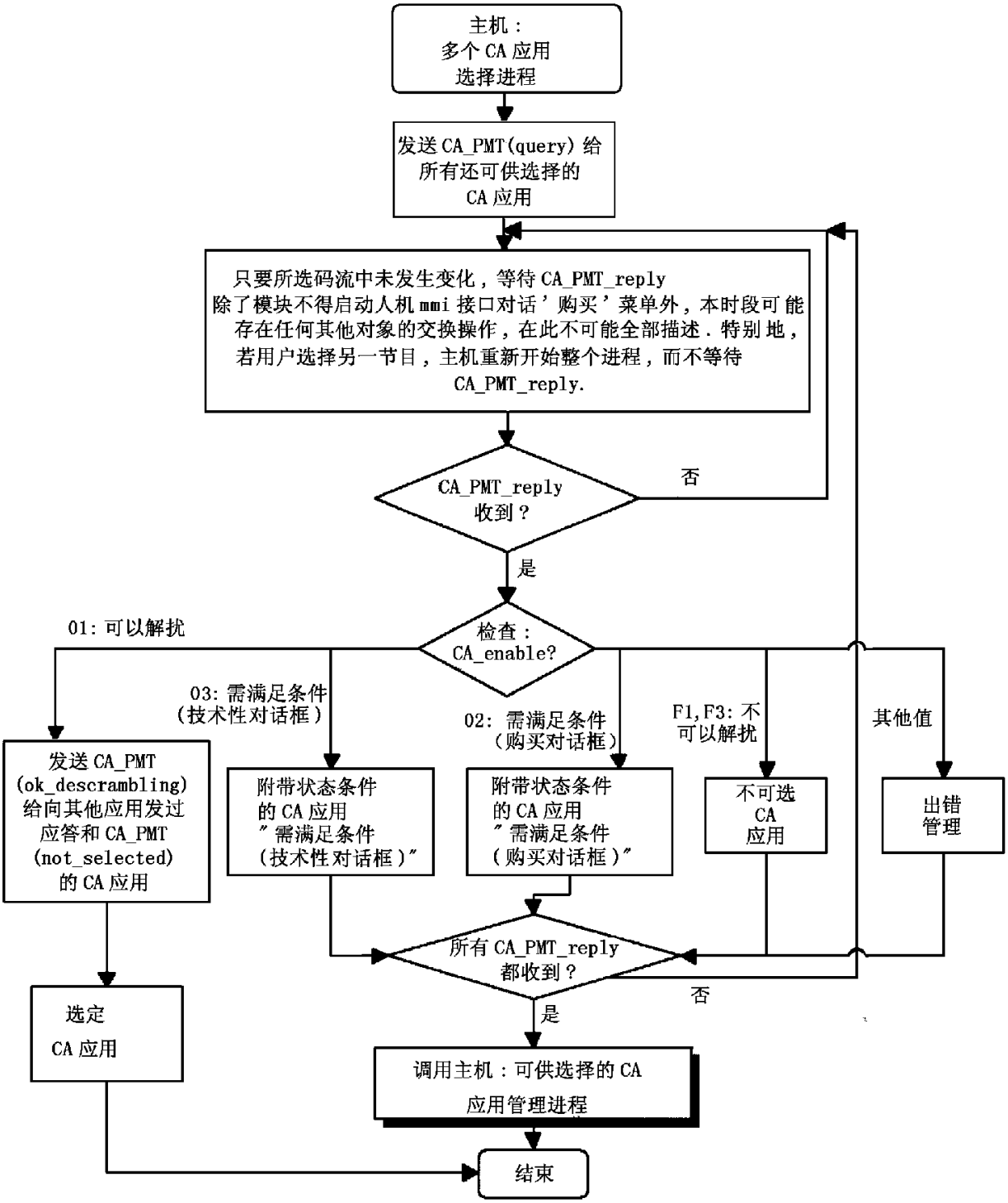


图 G9　CA _ PMT **和** CA _ PMT _ reply **使用的流程图之四**

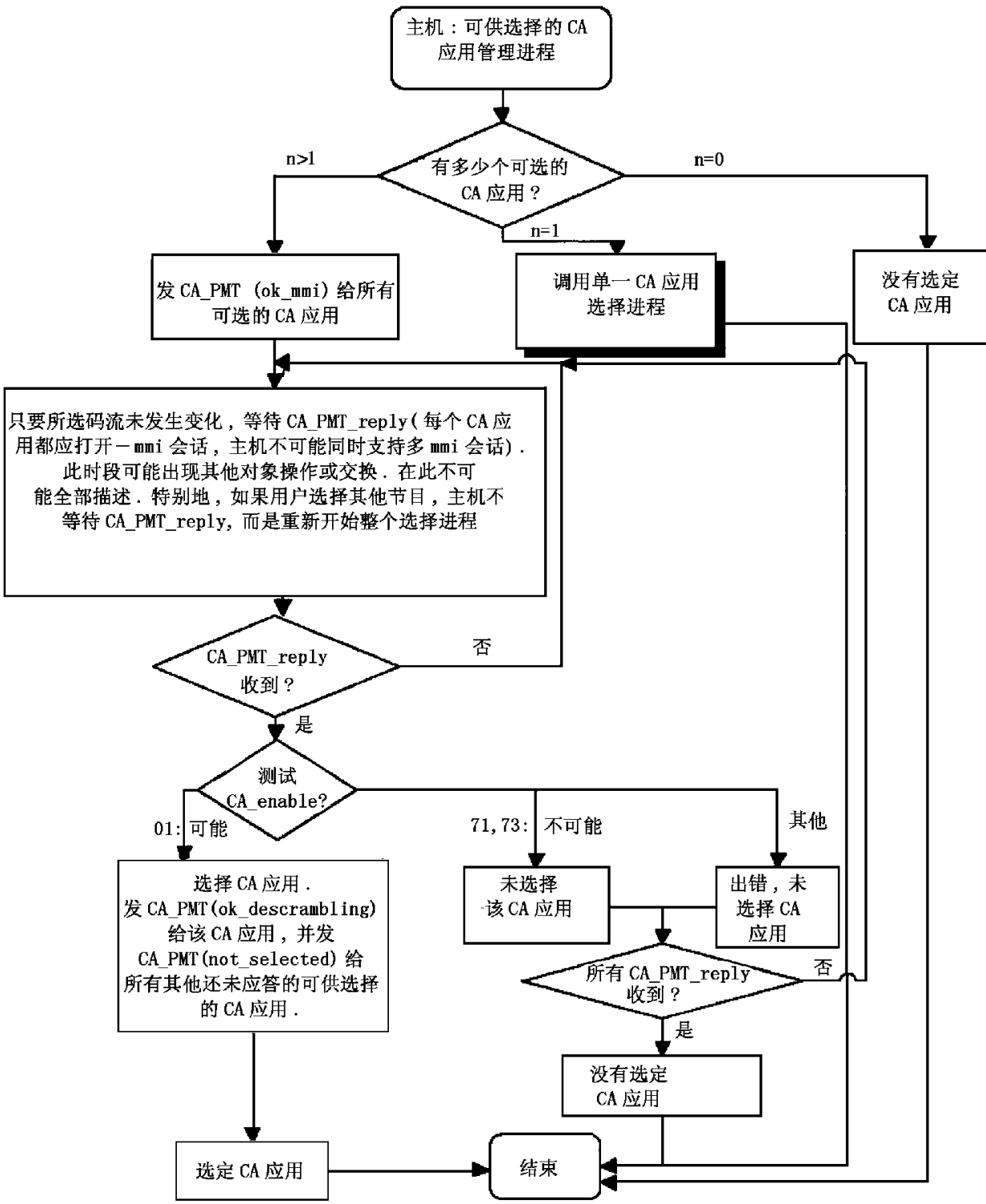


图 G10　CA _ PMT **和** CA _ PMT _ reply **使用的流程图之五**

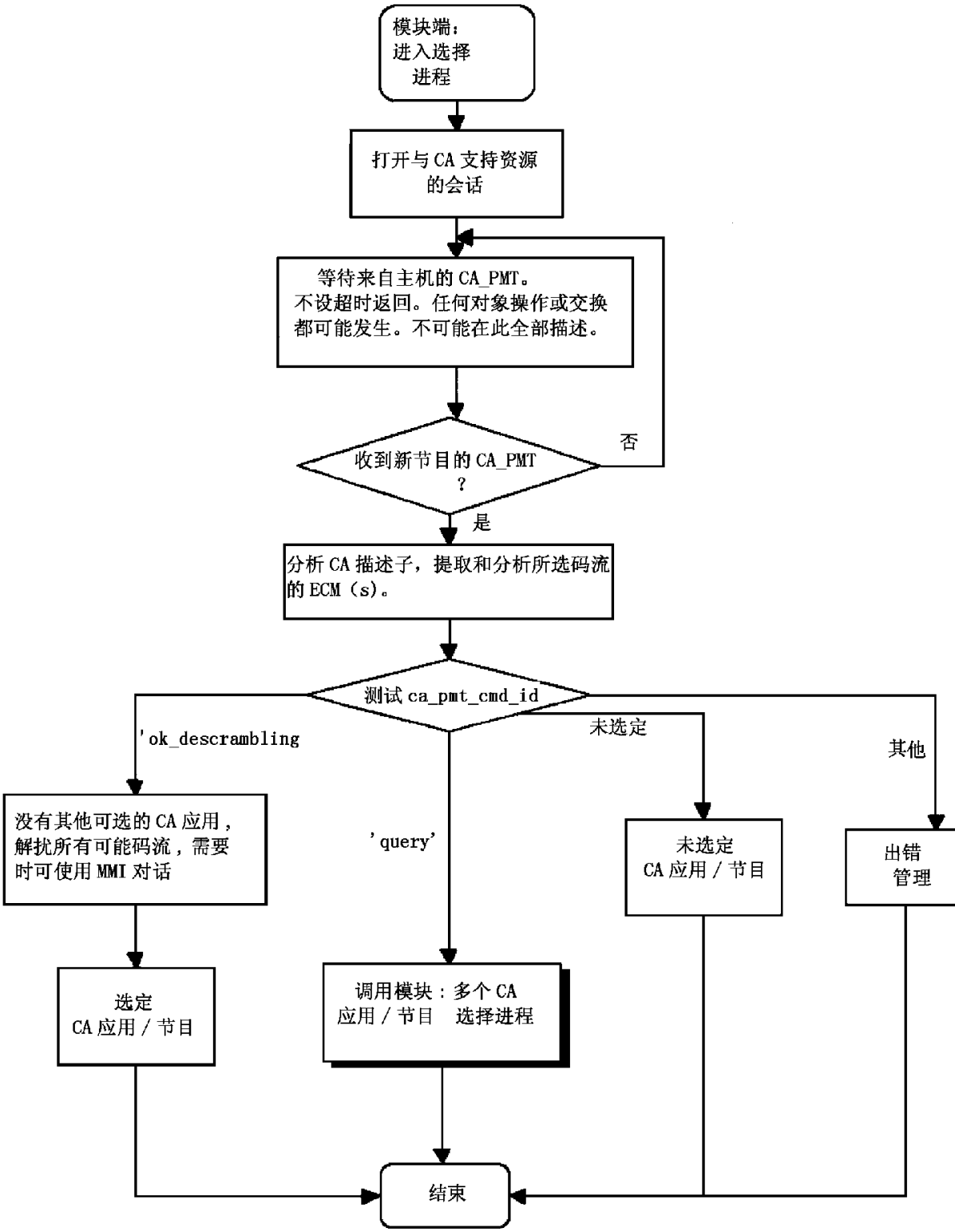


图 G11　CA _ PMT **和** CA _ PMT _ reply **使用的流程图之六**

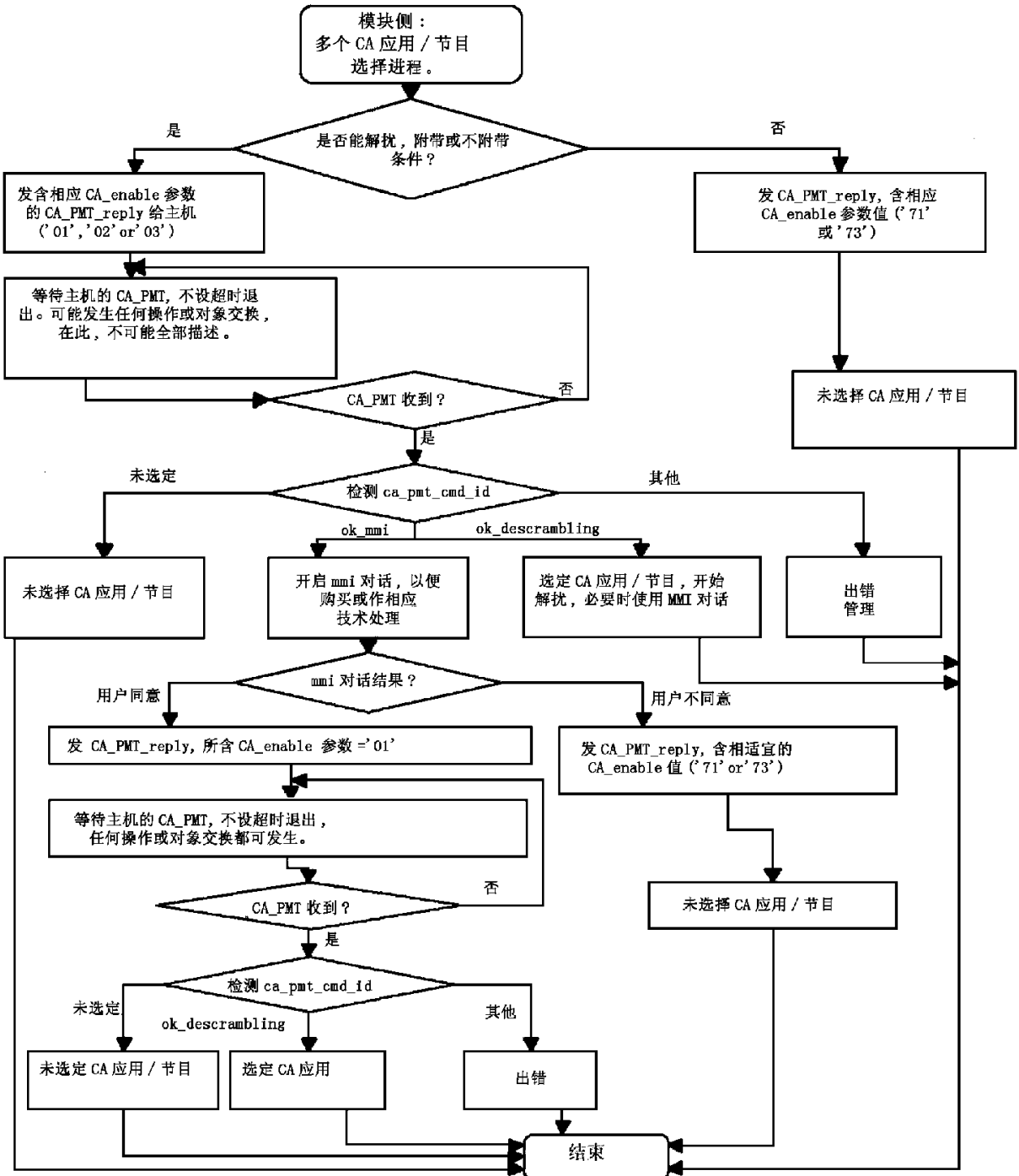


图 G12　CA _ PMT **和** CA _ PMT _ reply **使用的流程图之七**

G13　关于物理层死锁

下面是一个特定模块的单一缓冲区实现例子和对此实现可免于死锁的论证。但不保证此特定实现是稳健的或确实完全能用，它纯粹是个示例，使用单个缓冲区，当且仅当 FR＝＝1 **时，该缓冲区可用。模块卡实现** 3 **比特标志位：**DA，FR **和** FRx。**主机只能访问** DA **和** FR。FR **和** FRx **读写规则如下：**

模块若设置位 FRx，FR 和 FRx 将一起被置位；

若 FRx 被模块复位，或因主机写首字节而被复位，则 FRx 和 FR 一起复位；

如 FR 由模块复位，则只复位 FR，FRx 保持以前状态；

如 FRx 被复位，则模块不能置位 FR。

下列讨论考察了主机和模块争用空闲缓冲区（即 FR==1），主机和模块操作的所有可能顺序。对此过程中关键的时间点标示如下：(a1)(a2)(b1)等。

主机到模块协议：

(a1) 主机置 HC 为 1

(a2) 主机检测 FR

若 FR==0，(a3) 主机置 HC=0 结束（缓冲区正被占用）；

若 FR==1，主机可继续，写大小，字节。(a4) FR，FRx 在写首字节时被复位为 0。传输结束，(a5) 复位 HC=0 结束。

模块到主机协议：

(b1) 模块复位 FR

(b2) 模块检测 FRx

若 FRx==0，模块等待，缓冲区正被用；

若 FRx==1，(b3) 模块检测 HC；

若 HC==1，(b4) 模块置位 FRx 并等待（若 FRx 被复位，即主机正写缓冲区，置位操作将失败）；

若 HC==0，模块可继续，模块复位 FRx，填充缓冲区并 (b5) 置位 DA，结束。

若模块想启动一轮模块到主机的协议，有以下几种情形：

a) 模块在 a1 之前复位 FR 并检测 HC，查到 FRx==1 且 HC==0，模块可继续，主机查到 FR==0，复位 HC 并等待一物理查询周期。随着模块往前执行，主机须定期检查 DA 并在 FR 重新置位前读出数据。

模块在 a2 之前，复位 FR 且在 a1 之后检测 HC，模块看到 FRx==1 和 HC==1，尝试置位 FR 并等待，主机方面有两种可能，若模块在 a2 之前置位 FR 则主机将继续执行，否则主机会检测到 FR==0，复位 HC 并等待一物理查询周期。这种情形下若模块查询周期长于主机复位 HC 所需时间，则模块将在下一周期继续执行，否则模块可能等待一或两个模块查询周期。若模块由主机复位 HC 触发，则一旦发生主机复位 HC，模块就开始执行，模块将先执行而主机须等待 DA 被置位并在 FR 被重置位为 1 之前读出数据，若模块查询周期比主机的长，则主机将先开始执行；

b) 模块在 a2 之后检测 FR，在 a4 之前测 FRx，看到 FRx==1 则模块须在 a5 前检测 HC，此时 HC==1。模块试图置位 FR 并等待一个自身的查询周期。主机将检测到 FR==1 并继续。模块卡需检测缓冲区已满并在开始下一轮新的操作前读出数据。要求模块在 a5 之前必须检测 HC 并不麻烦，因为它已在 a4 之前检测过 FRx，故而有一个完整的写缓冲区周期（至少 3 字节）完成此工作；

c) 模块在 a4 之后检测 FRx，看到 FRx==0 并等待其查询周期，而主机写缓冲周期正在进行。模块需检测缓冲区已满并读出数据才能开始新一轮周期。不会出现其他情形。从主机方面看，情形或多或少象以上情形的翻版。有以下几种可能：

1) 主机置位 HC，在 b1 之前检测 FR，在进行到 a4。主机看到 FR==1 继续操作。模块看到

FRx==0，等待查询周期（等价于上述（d）情形）；

2）主机置位HC在b1前检测FR，主机看到FR==1继续操作。模块看到FRx==1和HC==1并等待一查询周期（等价于上述情形（c））；

3）在b3前主机置位HC，在b1之后测FR位，看到FR==0，复位HC并等待一物理层查询周期。模块看到FRx==1，若模块在主机复位HC前检测HC位，则它将等待一查询周期，若在主机复位HC之后检测HC位，则可继续操作（等价于上述情形（b））；

4）主机在b3后置位HC，主机看到FR==0，复位HC并等待一物理层查询周期。模块看到HC==0继续操作（等价于上述情形（a））。只要是任一方获得缓冲区访问权，则FR==0且一直保持到缓冲区清空。这种情形下会出现以下几种可能：

—主机获得访问权。FR，FRx在写头字节时复位，主机完成写缓冲区后复位HC，这种状态下，DA，FRx和HC全为0。模块识别到此种“主机已完成写”状态因而进行读空缓冲区操作，完成后置FR，FRx为1。若主机在此过程中置HC并检测FR，则FR==0故主机不能操作。若正好模块检测时HC碰巧为1，则模块只须等待一个模块查询周期。若在模块清空缓冲区时（读完）主机正在进行查询，模块将置位FR。主机可能看到或看不到这一设置，但不会死锁；

—模块获得缓冲区访问权。模块复位FRx，写缓冲区并置位DA。主机将在查询周期检测DA随后读出数据。DA在头字节读出后复位，主机继续完成从缓冲区中读空数据。模块检测这一过程并置位FR，FRx。重要的是主机要发数时检测FR位的同时检测DA位，因为单个缓冲区被模块填充之后将一直处于被占用状态，即DA为1，FR为0，直到主机清空缓冲区。

附 录 H
（文件的附录）
通用加扰系统

本附录说明如何在 GB/T 17975.1 规定的码流中进行加扰，描述了在不同的 CA 系统之间可以互操作所必须共同遵循的最基本条件：通用的加扰算法。每个用户接收设备中应集成相应的解扰模块。

H1 加扰算法

本部分包括对加扰算法的概述以及一些实现的条款。当在 TS 级别上进行加扰时，加扰算法对 TS 流的净荷进行加扰。

当采用同样加扰算法实现 PES 级别的加扰时，需要使用一种 PES 包结构。PES 级别的加扰算法要求 PES 包头不被加扰（如 GB/T 17975.1 中的规定），包含部分 PES 包的 TS 包不应包含 AF（当 TS 包包含 PES 包的结尾时例外）。已加扰的 PES 包的包头不应跨越多个 TS 包。载有已加扰的 PES 包开头的 TS 包用 PES 包头、PES 包净荷的第一部分填充。这样，PES 包净荷的第一部分实际作为一个大小基本相同的 TS 包净荷被加扰。剩下的 PES 包净荷被划分成大小为 184 字节的块。每个块和具有 184 字节的 TS 包的加扰的算法相同。PES 包净荷的结尾和 TS 包的结尾通过填充合适大小的 AF 来对齐（如 GB/T 17975.1 中的规定）。如果 PES 包的长度不是 184 字节的整数倍，PES 包净荷的最后部分（从 1～183 字节）的加扰与具有相同大小净荷的 TS 包的加扰算法相同。

图 H1 给出了加扰的 PES 包到 TS 包的映射方法：

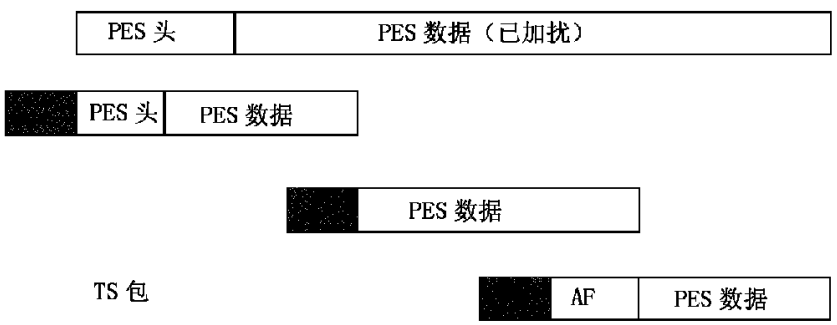


图 H1 PES 级别的加扰算法

为了使得解扰更简单，PES 级别的加扰算法对复接过程提出了一些限制。下面 H2.3 将给出把 PES 加扰包影射到 TS 包的建议方法。如果 TS 包加载 PES 包净荷时需要插入 AF，则这种方法将增加一些比特率的开销。在这种情况下，需要插入一个只包含一个 AF 的 TS 包。

当加扰用于 MPEG-2 时，由于 MPEG-2 句法中未包含加扰控制比特，会遇到下面的一个问题。对段（Section）的加扰不能在 TS 级别进行，而应该由加扰控制域比特来指示。在同一个 TS 包中不应该同时包含已加扰和未加扰的段。MPEG-2 定义的比特填充机制可以用来标志只包含未加扰段或已加扰段的 TS 包。承载段的 TS 包的结尾应由 0xff 值填充，确保已加扰和未加扰的段分离到不同的 TS 包中去。

通过增加加扰过程的复杂度，可以降低解扰器中的内存、解扰电路的成本。根据实现的不同，

解扰需要的内存、解扰延时也有所不同。

H2 在 MPEG-2 环境下使用加扰算法

本部分主要包含如何在 MPEG-2 比特流上有效使用加扰算法的句法、以及一部分实现方法。

H2.1 加扰控制域

在 TS 包头、PES 包头中，MPEG-2 系统包括了 2 比特的加扰控制域。这两个比特的含义在 MPEG-2 系统中只定义了一部分。

表 H1 给出了 TS 包中加扰控制比特的详细定义。

表 H1 TS 包加扰控制值

比特值	描述
00	对 TS 包不加扰（兼容 MPEG-2）
01	预留为将来用
10	采用偶密钥加扰
11	采用奇密钥加扰

第一个加扰控制比特指示净荷是否被加扰，第二个比特指示是采用偶还是奇密钥加扰。如果 TS 包的净荷不在 TS 级别加扰，在 PES 级别仍可被加扰。表 H2 给出了在 PES 包头中的加扰控制比特含义。由于在 PES 级和 TS 级上加扰控制比特和加扰算法的相似，使得解扰更容易。

表 H2 PES 包加扰控制值

比特值	描述
00	对 PES 包不加扰（兼容 MPEG-2）
01	预留为将来用
10	采用偶密钥加扰
11	采用奇密钥加扰

H2.2 CA 系统 ID 的注册

为标志不同的 CA 系统，在 MPEG-2 的 CA _ descriptor（）中的 CA _ System _ ID 需要注册。CA _ System _ ID 域使得数字电视接收机可以很容易地过滤相关的 CA 信息。

H2.3 PES 级加扰

为在广播体系下具有最大的操作灵活性，必须允许在 PES 级别进行加扰。为了避免用户接收端的设备进行复杂的处理，需要一个简单的解扰电路。本节中给出了对 PES 加扰的一些额外约束条件，以降低实现 PES 加扰的代价。这些约束不适用于未加扰的 PES 包、TS 级别的加扰。

建议 1：加扰只在 TS 或 PES 级别进行，而不应同时进行。

建议 2：以加扰的 PES 包的包头不应超出 184 字节。

建议 3：包含加扰的 PES 包一部分的 TS 包，除非是包含 PES 包的结尾，否则不应包含 AF。包含 PES 包的结尾的 TS 包，可以包含 AF，以对齐 PES 包和 TS 包的结尾。

H3 跨越媒体广播边界时的转移控制

MPEG-2 的 PSI 部分描述了如何寻找 CA 系统信息的句法。CA 表格、PMT 包含 CA 描述符，其中 CA _ PID 代表了包含 CA 系统如 ECM、EMM 的 TS 包的 PID。在跨越媒体广播边界时，可以把这些 TS 包中的 CA 信息使用其他 CA 系统代替。下面的约束使得载有 CA 信息的 TS 包的代替更加方便。

建议 4：所有 PID 和 CA _ descriptor 中 CA _ PID 相同的 TS 包只包含 CA 信息。其他任何地方不得包含 CA 信息。

建议 5：两个不同的 CA 系统供应商在相同 TS 流中不得包含相同的 CA _ PID 值。

这些建议已经足够实现在跨越媒体广播边界时滤出 CA 信息、使用新的 CA 信息取代。

H4 CA 数据

本部分描述了如 GB/T 17975.1 中规定了 CA 信息如 ECM、EMM、将来的授权信息等传输的段机制。CA 信息的构成根据每个 CA 系统供应商而不同。如表 H3 所示，采用了两个不同的 table _ id 值来区分两种不同类型的用于传输 ECM 的表。CA _ message _ section（）的头可以用于过滤。GB/T 17975.1中描述了如何在 TS 包中加载这些段。CA _ message _ sections 当插入到 TS 流中时，应被看成是 GB/T 17975.1 中的专有段。表 H3 中的 CA message 段最大长度为 256 字节。

表 H3 CA 信息表 CMT 的句法

句法	比特数	类型
CA _ message _ section () {		
table _ id	8	Uimsbf
section _ syntax _ indicator	1	Bslbf
reserved	1	Bslbf
ISO _ reserved	2	Bslbf
CA _ section _ length	12	Uimsbf
for (i=0; i<N; i++) {		
CA _ data _ Byte	8	Bslbf
}		
}		

CMT 的语义：

table _ id：见表 H4。

表 H4 Table ID 的分配

Table _ id 值	描述
0x00 - 0x02	MPEG 规定
0x03 - 0x3F	MPEG 预留
0x40 - 0x72	SI 规定
0x73 - 0x7F	预留
0x80	CA _ message _ section，ECM

表 H4 Table ID 的分配（完）

Table _ id 值	描述
0x81	CA _ message _ section，ECM
0x82 - 0x8F	CA _ message _ section，CA 系统专有
0x90 - 0xFE	专有
0xFF	ISO _ reserved

section _ syntax _ indicator：1 比特标志符，始终为 0。

reserved：表示预留为将来的应用使用，因此不能用做专有。

ISO _ reserved：表示预留为 ISO 将来的扩展定义使用。

CA _ section _ length：12 比特域，指示在 section _ length 域到段结尾的字节数。

CA _ data _ byte：载有专有 CA 信息的 8 比特域。前 17 个 CA _ data _ byte 字节可以用于地址过滤。

载有不同类型 CA 信息的 CA _ message _ sections 可以有 16 个 table _ id 的取值范围。其中两个 table _ id 域（0x80 和 0x81）预留为 ECM 数据用。当这两个 table _ id 值发生变化时，表示 ECM 内容发生了变化。这个变化条件可用于过滤 CA 信息。

附 录 I
（文件的附录）
与CA有关的业务信息/节目特定信息（SI/PSI）的规定

I1 SI/PSI表机制

I1.1 SI/PSI表PID取值

表I1给出了承载SI信息的TS包的PID值。

表I1 SI/PSI表的PID值

表	PID值
PAT	0x0000
CAT	0x0001
TSDT	0x0002
预留	0x0003～0x000F
NIT，ST	0x0010
SDT，BAT，ST	0x0011
EIT，ST	0x0012
RST，ST	0x0013
TDT，TOT，ST	0x0014
网络同步	0x0015
预留将来使用	0x0016～0x001B
带内信令	0x001C
测量	0x001D
DIT	0x001E
SIT	0x001F

I1.2 SI中关于加扰的描述

除含有时间表信息的EIT外，本附录中所有表格定义的信息都不加扰。

如果加扰是在TS层上进行，需要使用填充机制，以保证段的结尾和TS包的结尾对齐，这样加扰的数据、未加扰的数据可以在包的边界处分开。

为了标识用于控制EIT数据加扰的CA流，在PSI中需要标识加扰的EIT的时间计划数据。Service_id值0xFFFF用来标识加扰的EIT，这个服务的节目映射段中将把EIT描述成专有流，并应包含CA描述符。CA描述符给出了PID值以及其他专有数据来标识CA流。Service_id值0xFFFF不能用于其他业务。

I2 TS包头中与CA的相关部分

CA包括两种层次的加扰，PES层加扰和TS层加扰。TS层的加扰是基于TS传送流包的传输净荷进行的。而PES层的加扰是基于PES包结构。

在 MPEG-2 的系统规范中，TS 包头中有两比特的加扰控制（transport _ scrambling _ control）字段。字段位置如表 I2 所示：

表 I2 TS 包头格式

包头字段	长度（比特）	说明
Sync-Byte	8	=0x47；同步头
Transport-error-indicator	1	出错标志
Payload-unit-start-indicator（PUSI）	1	载荷中的指针域
Transport-priority	1	特别传输优先级规定
PID	13	包标识号
Transport-scrambling-control	2	加扰控制域
Adaptation-field-control	2	适配域控制
Continuity-counter	4	连续递增计数器

transport _ scrambling _ control 字段的意义在 MPEG-2 中只是部分被定义。本附录中给出了完整定义，如表 I3 所示。

表 I3 transport _ scrambling _ control 字段的意义

比特值	描述
00	TS 包净荷不加扰（MPEG 2 兼容）
01	预留将来使用
10	TS 包用偶密钥加扰
11	TS 包用奇密钥加扰

此处第一位的加扰控制位表示有效载荷是否加扰，第二位表示使用偶或奇密钥。

I3 CA 相关表格

与 CA 有关的表包括：CAT、PMT、CMT 等。

I3.1 条件接收表（CAT）

通过条件接收表可以为一个或多个条件接收系统与传送流中的 EMM 流等 CA 特殊参数建立关联。条件接收表中包含了 CA 描述符。

CAT 的 PID 取值为 0x0001。条件接收表的句法如表 I4 所示。

表 I4 CAT 表结构

句法	值	比特数	类型
Conditional _ access _ section（）{			
table _ id	0x01	8	uimsbf
Section _ syntax _ indicator	1	1	bslbf
0		1	bslbf

表 I4 CAT 表结构（完）

句法	值	比特数	类型
Reserved		2	bslbf
Section _ length		12	uimsbf
Reserved		18	bslbf
Version _ number		5	uimsbf
Current _ next _ indicator		1	bslbf
Section _ number		8	uimsbf
last _ section _ number		8	uimsbf
for (i=0; i<N; i++) {			
Descriptor ()			
}			
CRC _ 32		32	rpchof
}			

语义说明：

— **表格标识符**（table _ id）：**取值为** 0x01；

— **段句法指标**（section _ syntax _ indicator）：**段句法指标是一个设置为** 1 **的** 1 **比特字段**；

— **段长度**（section _ length）：**这是一个** 12 **比特的字段。它指出了段的字节数，从紧跟着** section _ length **的字段开始计算，包括** CRC **字段。段长度不应超过** 1 012（0x3FD）；

— **版本号**（version _ number）：**当** CAT **中所传送的信息发生改变时，版本号码应增值** 1。**当增值至** 31 **时，它返回至** 0。**当** current _ next _ indicator **被设置为** 1 **时，那么版本号应该是当前适用的** CAT **的版本号。当** current _ next _ indicator **被设置为** 0 **时，那么版本号码应该是下一个适用的** CAT **的版本号**；

— **当前下一个指针**（current _ next _ indicator）：**这是一个** 1 **比特的指标。当设置为** 1 **时，表明** CAT **是当前适用的。当字符设置为** 0 **时，表明所传送的** CAT **还不适用，而应该是下一个** CAT **才是有效的**；

— **段号**（section _ number）：**这** 8 **比特字段给出了段的号码。子表格中首段的号码应是**"0X00"。**每增加一个段，段号码加** 1；

— **最后段号**（last _ section _ number）：**这** 8 **比特的字段规定了子表格中最后一段的号码，即带有最高号码的段**；

— **描述符**（Descriptor）：CAT **中的描述符包括** CA _ descriptor（）**及专有** descriptor。**其中** CA _ descriptor **中的** CA _ PID **指向传送流中的** EMM **流**。

I3.2 节目映射表（PMT）

节目映射表中也包含了 CA **描述符，其中的** CA _ PID **字段可作为** TS **包** PID **值的参考，用来传递** CA **信息，如** EMM、ECM **等。用广播分配媒体边界上的其他** CA **数据来替代这些** TS **包中的（部分）** CA **信息是可行的。**

I3.3 CA 消息表（CMT）

这些CA信息包括ECM、EMM以及未来的授权管理数据。这些CA信息的结构与具体的CA系统相关。

ECM(Entitlement Control Message)即授权控制信息,它属于与基本流相关的CA管理信息。ECM中传送与码流控制字有关的加密信息。

EMM(Entitlement Management Message)即授权管理信息,它属于系统级的条件接收管理信息。EMM中与用户权限有关的加密信息。

ECM、EMM的句法均遵从CA_message_section()的句法定义。CA_message_section()的头部能够用于ECM、EMM的过滤。CA_message_section()的具体句法如表I5所示。

表I5 CMT表结构

句法	比特数	类型
CA_message_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved	3	bslbf
CA_section_length	12	uimsbf
for(i=0; i<N; i++){		
CA_data_byte	8	bslbf
}		
}		

语义说明:

— 表格标识符(table_id):取值见表I6。

表I6 table_id取值

Table_id值	描述
0x00 — 0x02	参见GY/Z 174中表2
0x03 — 0x3F	参见GY/Z 174中表2
0x40 — 0x73	参见GY/Z 174中表2
0x74 — 0x7D	参见GY/Z 174中表2
0x7E	参见GY/Z 174中表2
0x7F	参见GY/Z 174中表2
0x80	CA_message_section,ECM
0x81	CA_message_section,ECM
0x82 — 0x8F	CA_message_section,CA系统专有
0x90 — 0xFE	参见GY/Z 174中表2
0xFF	参见GY/Z 174中表2

— 段句法指示符(section_syntax_indicator):1比特的指示符,置为0;

— CA 段长度（CA _ section _ length）：12 比特字段，表示从 section _ legnth 字段直至段结束的字节数；

— CA 数据字段：以 8 比特为单位的字段，用于传送专有 CA 信息，前 17 个 CA _ data _ Byte 可用于地址过滤。

在 CA _ message _ section 中，共有 16 种 table _ id 取值（0x80 到 0x8F），用于携带不同的条件接收信息。table _ id 中的两个值，0x80 和 0x81，预留给 ECM 数据的传送。这两个 table _ id 值的变化指示了 ECM 内容有所变化，该变化条件可用于过滤条件接收信息。

I3.4 free _ ca _ mode

I3.4.1 SDT 表中的 free _ ca _ mode

SDT 表句法见表 I7。

表 I7 SDT 表句法

句法	值	比特数	类型
Service _ description _ section () {			
table _ id	0x42/0x46	8	Uimsbf
Section _ syntax _ indicator	1	1	Bslbf
reserved _ future _ use		1	Bslbf
Reserved		2	Bslbf
Section _ length		12	Uimsbf
Transport _ stream _ id		16	Uimsbf
Reserved		2	Bslbf
Version _ number		5	Uimsbf
Current _ next _ indicator	0/1	1	Bslbf
Section _ number		8	Uimsbf
last _ section _ number		8	Uimsbf
Original _ network _ id		16	Uimsbf
reserved _ future _ use		8	Bslbf
for (i=0; i<N; i++) {			
service _ id		16	Uimsbf
Reserved _ future _ use		6	Bslbf
EIT _ schedule _ flag		1	Bslbf
EIT _ present _ following _ flag		1	Bslbf
running _ status		3	Uimsbf
free _ CA _ mode		1	Bslbf
descriptors _ loop _ length		12	Uimsbf
for (j=0; j<N; j++) {			

表 I7 SDT 表句法（完）

句法	值	比特数	类型
descriptor ()			
}			
}			
CRC _ 32		32	Rpchof
}			

语义说明：

— free _ CA _ mode：**这是个** 1 **比特的域，当为** 0 **时表示服务的所有部件的流未加扰；当为** 1 **时，表示接收** 1 **路或多路码流时需要** CA **系统控制**。

I3.4.2 EIT **表中的** free _ ca _ mode

EIT **句法见表** I8。

表 I8 EIT **句法**

句法	值	比特数	类型
service _ association _ section () {			
table _ id	0x4E～0x4F 0x50～0x5F 0x60～0x6F	8	uimsbf
section _ syntax _ indicator		1	bslbf
Reserved _ future _ use		1	bslbf
Reserved		2	bslbf
section _ length		12	uimsbf
service _ id		16	uimsbf
Reserved		2	bslbf
version _ number		5	uimsbf
current _ next _ indicator	0/1	1	bslbf
section _ number		8	uimsbf
last _ section _ number		8	uimsbf
Transport _ stream _ id		16	uimsbf
original _ network _ id		16	uimsbf
last _ table _ id		8	uimsbf
Segment _ last _ section _ number	s_0+n-1	8	uimsbf
for (i=0; i<N; i++) {			
Event _ id		16	uimsbf

表 I8 EIT 句法（完）

句法	值	比特数	类型
Start _ time		40	bslbf
Duration		24	uimsbf
running _ status		3	uimsbf
free _ CA _ mode		1	bslbf
Descriptors _ loop _ length		12	uimsbf
for (j=0; j<N; j++) {			
Descriptor ()			
}			
}			
CRC _ 32		32	rpchof
}			

语义说明：

— free _ CA _ mode：**这是个** 1 **比特的域，当为** 0 **时表示事件的所有相关的流未加扰；当为** 1 **时，表示接收其中的** 1 **或多个流时需要** CA **系统。**

I4 相关描述符

与 CA **相关的描述符有** CA _ descriptor ()、CA _ identifier _ descriptor () **及** CA _ system _ descriptor ()。**其中** CA _ descriptor () **在** GB/T 17975.1 **中定义，** CA _ identifier _ descriptor () **及** CA _ system _ descriptor () **在** GY/Z 174 **中规定。**

表 I9 **为这三个描述符在** PSI/SI **表中出现的可能位置。** * **表示可能出现，**—**表示不可能出现。**

表 I9 CA 相关描述符在 PSI/SI 中的可能位置

描述符	标签值	PMT	CAT	NIT	BAT	SDT	EIT	OT	PMT	SIT
CA _ descriptor	0x09	*	*	—	—	—	—	—	—	*
CA _ identifier _ descriptor	0x53	—	—	—	*	*	*	—	—	*
CA _ system _ descriptor	0x65	—	—	—	—	—	—	—	*	—
注：SIT **只出现在部分传送流中。**										

I4.1 CA _ descriptor

条件访问描述符用于描述 EMM **和** ECM。**当任何基本流被加扰时，条件访问描述符必须在基本流所在的** PMT **中出现。当任何与传送流相关的系统级** CA **管理信息存在时，条件访问描述符必须在** CAT **中出现。**

CA _ descriptor **的句法定义见表** I10。

表 I10 CA _ descriptor 的句法

句法	比特数	类型
CA _ descriptor () {		
Descriptor _ tag	8	uimsbf
Descriptor _ length	8	uimsbf
CA _ system _ id	16	uimsbf
Reserved	3	bslbf
CA _ PID	13	uimsbf
for (i=0; i<N; i++) {		
private _ data _ byte	8	uimsbf
}		
}		

语义说明：

— 描述符标记（descriptor _ tag）：取值为 0x09；

— 描述符长度（descriptor _ length）：表示从紧跟着 descriptor _ length 的字段直至描述符结束的总字节数；

— CA 系统标识符（CA _ system _ id）：表明提供 ECM、EMM 等条件接收信息的 CA 系统类型；

— CA 信息 PID（CA _ PID）：表示包含 ECM、EMM 信息的传送流的 PID 值。CA _ PID 所在的表不同，CA _ PID 所指示的内容则不同。当 CA _ descriptor 出现在 CAT 中时，CA _ PID 指向传送流中的 EMM 流。当 CA _ descriptor 出现在 PMT 中时，CA _ PID 指向传送流中的 ECM 流。

注：

1 PID 值与 CA _ descriptor 给出的 CA _ PID 值相等的所有 TS 包应只含有 CA 系统信息，其他地方不能带有 CA 信息（例如：适配域）。

2 两个不同的 CA 系统供应商在同一 TS 里不能有共同的 CA _ PID 值。

I4.2 CA _ identifier _ descriptor

CA 标识描述符表明了一个特殊的业务群、业务或事件与一个条件接收系统相关联，并以 CA _ system _ id（CA 系统标识符）标识 CA 系统的类型。

如果一个服务是采用条件访问保护的，则可用此描述符传送 CA 系统的数据。此描述符并不包含在任何 CA 控制功能中，它用于接收机的用户界面软件中，它能指出某种业务是 CA 控制的，然后用户接口软件可以决定这个业务可否实现。这样可以避免用户费心选择一个服务而此服务对用户并不可用。此描述符的播发在 SDT 中是可选的，在一描述符循环中它仅能出现一次。

CA _ identifier _ descriptor 的句法定义见表 I11。

表 I11 CA _ identifier _ descriptor 的句法

句法	比特数	类型
CA _ identifier _ descriptor () {		
descriptor _ tag	8	uimsbf
descriptor _ length	8	uimsbf
for (I=0; i<N; i++) {		
CA _ system _ id	16	uimsbf
}		
}		

语义说明：

— **描述符标记**（descriptor _ tag）：**取值为** 0x53；

— **描述符长度**（descriptor _ length）：**表示从紧跟着** descriptor _ length **的字段直至描述符结束的总字节数**；

— CA **系统标识符**（CA _ System _ ID）：**表明** CA **系统的类型**。

I4.3 CA _ system _ descriptor

CA _ system _ descriptor () **本附录未定义**。