

高速公路区域联网不停车收费示范工程暂行技术要求 第 10 部分

OBU 初始化编程器技术要求

2008 年 8 月

目 录

1 范围.....	1
2 引用文件.....	1
3 术语定义.....	1
4 缩略术语.....	2
5 编程系统构成.....	2
6 设备技术要求.....	3
6.1 微波通信.....	3
6.2 数据通讯.....	3
6.3 IC 卡读写（可选）	4
6.4 图像处理（可选）	4
6.5 数据存储（可选）	4
6.6 环境适应性.....	4
6.7 安全.....	4
6.8 可靠性.....	4
6.9 电池工作时间.....	4
6.10 发行成功率.....	4
6.11 一次发行时间.....	4
6.12 输入界面（可选）	4
7 应用规范.....	4
7.1 一次发行.....	5
7.2 二次发行.....	5
8 编程接口规范.....	5
8.1 编程器与 OBU 接口规范	5
8.2 编程器与计算机物理接口规范.....	6
8.3 编程器与计算机应用编程接口.....	7

1 范围

本标准规定了电子收费专用短程通信车载设备编程系统的构成、设备技术要求、应用规范，编程接口规范。

本部分适用于公路电子收费系统领域车载设备的初始化以及个性化。

2 引用文件

下列文件中的条款通过本部分的引用而成为本部分的条款。凡是注日期的引用文件，其随后所有的修改单（不包括勘误的内容）或修订版均不适用于本部分，然而，鼓励根据本部分达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本部分。

GB/T 20839-2007 智能运输系统 通用术语

GB/T 20851.1-2007 电子收费 专用短程通信 第1部分：物理层

GB/T 20851.2-2007 电子收费 专用短程通信 第2部分：数据链路层

GB/T 20851.3-2007 电子收费 专用短程通信 第3部分：应用层

GB/T 20851.4-2007 电子收费 专用短程通信 第4部分：设备应用

JR/T 0025-2005 中国金融集成电路（IC）卡规范

3 术语定义

GB/T 9387.1-1998 和 GB/T 20839-2007（《智能运输系统 通用术语》）确立的以及下列术语和定义适用于本部分。

3.1 应用 application

DSRC协议服务的用户。

3.2 文件 file

车载设备（OBU）应用数据的基础组织单位，一般多个相关的数据单元组成一个文件。

3.3 文件标识 file identifier

文件的标识号码，同一目录下，文件标识号是唯一的。

3.4 编程 programming

初始化车载设备的密钥、系统信息，个性化车载设备的车辆信息，也称为发行。

3.5 联机 online

编程器与计算机连接。连接包括有限连接和无线连接。

3.6 脱机 offline

编程器不与计算机连接。连接包括有限连接和无线连接。

4 缩略术语

下列缩略语适用于本部分。

ADU 应用数据单元 (Application Data Unit)

AID 应用标识 (Application Identifier)

APDU 应用协议数据单元 (Application Protocol Data Unit)

ASDU 应用服务数据单元 (Application Service Data Unit)

ASN.1 抽象语法记法一 (Abstract Syntax Notation One)

BST 信标服务表 (Beacon Service Table)

DID 目录标识 (Directory Identifier)

DSRC 专用短程通信 (Dedicated Short Range Communication)

ETC 电子收费 (Electronic Toll Collection)

FID 文件标识 (File Identifier)

IID 启用标识 (Invoker Identifier)

MAC 媒体访问控制 (Medium Access Control)

OBU 车载设备 (On Board Unit)

RID 记录标识 (Record Identifier)

RSU 路侧单元 (Roadside Unit)

5 编程系统构成

车载设备编程系统的总体架构如图1所示：

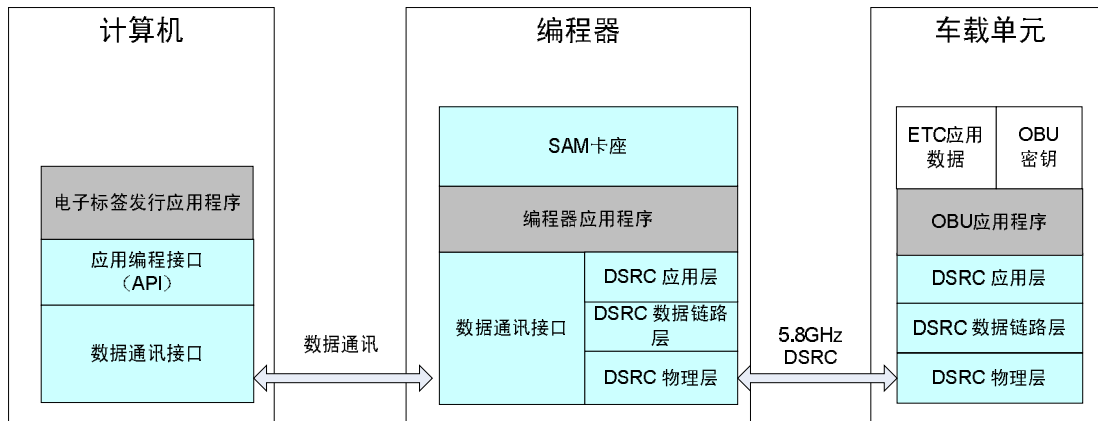


图1 系统构成

说明：

上图中，视应用场合不同，计算机部分为可选项。比如脱机应用时，可以不用计算机。系统构成分为3个部分：计算机、编程器、车载设备，各个部分的功能和接口如表1所示：

表1 车载设备编程系统构成表

构成部分	功能职责	连接接口	连接对象
计算机	负责组织数据，发起并控制编程流程。主要以编程API形式提供给编程程序调用。	1: TCP/IP、RS232接口，USB接口等。	编程器
编程器	接收并计算机指令，完成编程过程。	1: TCP/IP、RS232接口，USB接口等 2: 5.8GHz无线链路。	计算机、车载设备
车载设备	接收并执行编程器指令，存储编程数据。	1: 5.8GHz无线链路。	编程器

6 设备技术要求

本规范只对编程器提出技术要求，其他部分参照相关标准。

6.1 微波通信

物理层、数据链路层、应用层、设备应用应分别符合国标GB/T20851.1-2007、GB/T20851.2-2007、GB/T20851.3-2007、GB/T20851.4-2007的规定。

6.2 数据通讯

编程器与PC的接口应支持但不限于TCP/IP、RS232接口、USB接口，可选一项或多项。

6.3 IC卡读写（可选）

6.3.1 接触式读写

卡座数量： ≥ 2 个。符合ISO7816协议。支持T=0，T=1两种协议。

6.3.2 非接触式读写

符合ISO14443 TYPE A协议。支持逻辑加密卡，双界面CPU卡。

6.4 图像处理（可选）

支持图象抓拍、存储。

6.5 数据存储（可选）

支持数据存储功能，支持记录、文件格式。

6.6 环境适应性

工作温度： $-20^{\circ}\text{C} \sim +55^{\circ}\text{C}$ 。

存储温度： $-20^{\circ}\text{C} \sim +55^{\circ}\text{C}$ 。

相对工作湿度： $10\% \sim 90\%$ 。

6.7 安全

符合GB 4943 – 1995信息技术设备的安全的规定。

6.8 可靠性

$\text{MTBF} \geq 5000 \text{ (h)}$ 。

6.9 电池工作时间

电池充满电后，连续工作时间 ≥ 6 小时，支持发行300个车载单元。

6.10 发行成功率

$\geq 90\%$ 。

6.11 一次发行时间

≤ 15 秒。

6.12 输入界面（可选）

键盘、触摸屏。

7 应用规范

车载设备编程即建立或修改车载设备的初始化信息，包括密钥、系统信息文件、车辆信息文件、其他文件。

车载设备的信息的存取基于安全模块，车载设备编程也是基于安全模块进行，所有对密钥和文件的建立或修改必须符合安全模块的安全规范，密钥的定义，文件格式等必须符合相关的规范。

编程有两种方式：联机方式和脱机方式。

I 采用联机编程方式时由计算机控制发行流程，此时编程器作为一个通道，转发计算机与车载设备之间的数据通讯，数据输入，密钥计算，流程控制等工作由计算机完成。

- I 采用脱机编程方式时由编程器控制发行流程，数据输入/修改，密钥计算，流程控制等工作由编程器完成。

车载设备编程也称为发行，发行可分为一次发行和二次发行。

7.1 一次发行

一次发行在一个相对安全的地方如运营中心进行，主要功能是替换车载设备的初始密钥，修改系统信息文件。根据应用要求不同，也可以重建车载设备的密钥系统和文件系统并初始化密钥和系统信息文件。一般需要联机发行，发行所涉及的密钥计算可以使用加密机或母卡。

7.2 二次发行

二次发行在营业网点进行，或者上门安装，主要功能是将车辆信息等个性化信息写入车辆信息文件。根据应用要求不同，二次发行还可以对系统信息文件和车辆信息文件以外的其他文件进行修改。二次发行可以联机进行或者脱机进行，发行所涉及的密钥计算可以使用加密机或SAM卡，在脱机方式下发行时，只用SAM卡。

当二次发行使用脱机方式发行时，发行过程如下：

- a) 计算机通过编程接口，将需要发行的车辆信息以记录的方式保存到编程器的数据库，可以保存一条或多条。
- b) 计算机与编程器断开连接。
- c) 安装OBU。
- d) 用户通过编程器的操作界面选择需要发行的记录并开始编程。
- e) 编程器控制编程的流程，并更新数据库记录的状态。
- f) 发行完毕，抓拍车辆图片，并保存为文件，将数据库记录与文件相关联(可选)。
- g) 计算机与编程器重新连接。
- h) 计算机读取编程器的数据数据库，下载相关文件，与后台数据库保持同步。

8 编程接口规范

8.1 编程器与 OBU 接口规范

编程器与OBU接口规范规定了编程器于车载设备之间使用的服务原语。基于方便性、通用性、可扩展性考虑，编程器与车载设备之间的原语使用《GB/T 20851.3-2007 电子收费 专用短程通信 第3部分：应用层》和《GB/T 20851.4-2007 电子收费 专用短程通信 第4部分：设备应用》中服务原语Initialization和TransferChannel。

本章对《GB/T 20851.3-2007 电子收费 专用短程通信 第3部分：应用层》中附录A.2 DSRC

应用目录补充OBU发行应用(obu-issuing)这一项，补充后定义如下：

DsrcApplicationEntityID::=INTEGER{	
system	(0),
electronic-toll-collection	(1), -- 电子应用
road-tag-station-management	(2), -- 道路标识站应用
electronic-road-price	(3), --城市道路收费
freight-fleet-management	(4),
public-transport	(5),
traffic-traveller-information	(6),
traffic-control	(7),
parking-management	(8),
geographic-road-database	(9),
medium-range-preinformation	(10),
man-machine-interface	(11),
emergency-warning	(12)
obu-issuing	(13) --OBU发行应用
} (0..31,...)	

8.2 编程器与计算机物理接口规范

8.2.1 通信接口物理形式

RSU与PC之间的通信接口应至少支持下列几种方式之一：

Ø 标准串行接口。

可采用 RS-232、RS-485 等，要求通讯波特率至少达到 115200。建议设置：串行口采用半双工的异步串行通讯方式，协议格式为“115200,N,8,1”，即波特率 115200bps，无奇偶校验，8 位数据，1 个停止位。

Ø 以太网接口

采用以太网（RJ45 接口），及 TCP/IP 协议进行连接。

Ø USB 接口

应兼容 USB1.1 及 USB2.0。

8.2.2 通信数据帧格式

在各种接口应用模式及物理接口形式下，RSU和PC间通讯的数据帧格式均应满足如下数据帧结构的要求：



数据帧中各数据域的说明如下：

字段	描述
STX	帧开始标志（Start Of Frame），取值为 FFFFH；
RSCTL	数据帧序列号，1 个字节； (1) RSU 发送的数据帧序列号的低半字节为 8，高半字节一般为 0~7，RSU 上电时发送的数据帧序号高半字节为 9； (2) PC 发送的数据帧序列号是将收到的数据帧序号高低半字节互换；

	(3) RSU 发送的数据帧序号为 X8H, 其中 X 为 0, 1, 2, 3, 4, 5, 6, 7, 9; (4) PC 发送的数据帧序号为 8XH, 其中 X 为 0, 1, 2, 3, 4, 5, 6, 7, 9;
DATA	发送的数据;
BCC	异或校验值, 从 RSCTL 到 DATA 所有字节的异或值;
ETX	帧结束标志, 取值为 FFH。

说明: 当RSU与车道控制器之间采用标准串行接口时, 需要进行特殊字节转义处理。特殊字节转义处理如下:

数据帧开始标志和帧结束标志为FFH。其他字段不能出现FFH, 如果数据确实为FFH, 需对其进行转义处理。

发送数据时, 如果在其它字段中出现FFH字节时, 将FFH分解为FEH和01H这两个字节来发送; 如果在其它字段出现FEH字节时, 需将FEH分解为FEH和00H这两个字节来发送。

接收数据时, 如果出现“FE 01”这样连续两个字节时将之合为一个字节FFH; 如果出现“FE 00”这样连续两个字节时将之合为一个字节FEH。

8.3 编程器与计算机应用编程接口

编程接口指计算机的API接口, 这里定义的API接口以C语言定义。编程接口应该提供的接口函数如表3所示:

表3 编程接口函数列表

序号	函数功能	函数定义
1	初始化连接	int _stdcall ApiInit(char* InitParam);
2	释放连接	int _stdcall ApiRelease();
3	打开设备	int _stdcall DevOpen(int TxPower);
4	关闭设备	int _stdcall DevClose();
5	BST	int _stdcall Initialization_request(byte* BSTData, int DataLen);
6	VST	int _stdcall Initialization_response(byte* VSTData, int* DataLen, int Timeout);
7	通道请求	int _stdcall TransferChannel_request(byte* TransData, int DataLen);
8	通道返回	int _stdcall TransferChannel_response(byte* TransData, int* DataLen, int Timeout);
9	下载二次发行	int _stdcall PushEtcInfo(byte* ObuSerialNo, byte*

	ETC车辆信息文件到编程器	EtcInfo, int InfoLen);
10	计算机与编程器同步二次发行结果	int _stdcall SyncEtcInfo(byte* ObuSerialNo, int* NoCount);
11	从编程器读取二次发行结果数据	int _stdcall ReadEtcInfo(byte* ObuSerialNo, byte* status, byte* EtcInfoData, int InfoLen, byte* PicBuf, int* PicLen);

说明：

上表中函数9—11可根据发行方式或设备情况，为可选函数。

I 初始化连接

Ø 函数原型：

```
int _stdcall ApiInit(char* lpParam);
```

Ø 函数功能：

初始化连接，打开TCP，串口等资源，连接到编程器设备。

Ø 函数参数：

2 lpParam，连接参数，为TCP/IP连接时的参数，形式如TCP: 192.168.0.1,12345，COM: 1，USB: \usb1\xxx 等等，具体参数详细的定义由各厂商自行定义。

Ø 函数返回：

0，初始化成功，1，初始化失败。其他由各厂商定义。

I 释放连接

Ø 函数原型：

```
int _stdcall ApiRelease();
```

Ø 函数功能：

释放连接，以及连接所占用的资源。

Ø 函数参数：

无

Ø 函数返回：

0，释放成功，1，释放失败。其他由各厂商定义。

I 打开设备

Ø 函数原型:

```
int _stdcall DevOpen(int TxPower);
```

Ø 函数功能:

打开设备，设置微波参数。

Ø 函数参数:

2 TxPower, 功率，范围1—31。

Ø 函数返回:

0, 打开成功，1, 打开失败。其他由各厂商定义。

I 关闭设备

Ø 函数原型:

```
int _stdcall DevClose(int TxPower);
```

Ø 函数功能:

关闭设备。

Ø 函数参数:

无。

Ø 函数返回:

0, 关闭成功，1, 关闭失败。其他由各厂商定义。

I BST

Ø 函数原型:

```
int _stdcall Initialization_request(byte* BSTData, int DataLen);
```

Ø 函数功能:

发送BST，编程器开始寻找OBU，如果在一定时间内（各厂商字定）没有寻到OBU，则返回失败。

Ø 函数参数:

2 BSTData, 发送的BST原始数据内容，其中不包括帧起始/结束标志7E、帧校验FCS。

2 DataLen, BST数据内容的长度。

Ø 函数返回:

0, 发送成功。1, 发送失败。其他由各厂商定义。

Ø 说明:

调用Initialization_request函数后, 应立刻调用Initialization_request。

I VST

Ø 函数原型:

```
int _stdcall Initialization_response(byte* VSTData, int* DataLen, int  
Timeout);
```

Ø 函数功能:

接收BST返回的VST内容。

Ø 函数参数:

- ² VSTData, 接收缓冲区, 调用者应为缓冲分配足够长度。VSTData为VST原始数据内容, 其中不包括帧起始/结束标志7E、帧校验FCS。
- ² DataLen, 接收到的实际长度。
- ² Timeout, 超时时间, 单位毫秒。

Ø 函数返回:

0, 接收成功。1, 接收失败。2, 接收超时。其他由各厂商定义。

I 通道请求

Ø 函数原型:

```
int _stdcall TransferChannel_request(byte* TransData, int DataLen);
```

Ø 函数功能:

调用该函数时, 编程器将原始数据TransData转发给OBU。

Ø 函数参数:

- ² TransData, 转发的数据内容, 内容为服务原语的原始内容, 其中不包括帧起始/结束标志7E、帧校验FCS。。
- ² DataLen, 数据长度。

Ø 函数返回:

0, 发送成功。1, 发送失败。其他由各厂商定义

Ø 说明:

调用TransferChannel_request函数后, 应立刻调用TransferChannel_request。

I 通道返回

Ø 函数原型:

```
int _stdcall TransferChannel_response(byte* TransData, int* DataLen, int  
    Timeout);
```

Ø 函数功能:

接收通道返回的数据内容。

Ø 函数参数:

2 TransData, 接收缓冲区, 调用者应为缓冲分配足够长度。TransData为服务原语原始数据内容, 其中不包括帧起始/结束标志7E、帧校验FCS。

2 DataLen, 接收到的实际长度。

2 Timeout, 超时时间, 单位毫秒。

Ø 函数返回:

0, 接收成功。1, 通道操作失败。2, 接收超时。其他由各厂商定义。

I 下载二次发行ETC车辆信息文件到编程器

Ø 函数原型:

```
int _stdcall PushEtcInfo(byte* OBUSerialNo, byte* EtcInfo, int InfoLen);
```

Ø 函数功能:

将要二次发行的车辆信息以记录的形式保存到编程器。

Ø 函数参数:

2 OBUSerialNo, OBU应用序列号, 长度8字节, 二进制格式。

2 EtcInfo, 要保存的ETC应用文件内容, 二进制格式, 内容符合相关规范的规定。

2 InfoLen, EtcInfo内容长度。

Ø 函数返回:

0, 保存成功。1, 保存失败。其他由各厂商定义

Ø 说明:

一般情况下, 要求发行时车辆信息跟OBUSerialNo绑定, 避免错发。

I 计算机与编程器同步二次发行结果

Ø 函数原型:

```
int _stdcall SyncEtcInfo(byte* OBUSerialNo, int* NoCount);
```

Ø 函数功能:

读取编程器保存的二次发行记录数。

Ø 函数参数：

2 ObuSerialNo, OBU编号列表缓冲区，每个ObuSerialNo长度为8字节，因此缓冲区长度为8的倍数，调用者为缓冲区分配足够长度。

2 NoCount, ObuSerialNo列表个数。

Ø 函数返回：

0, 同步成功。1, 同步失败。其他由各厂商定义

Ø 说明：

调用SyncEtcInfo后应如果IdCount大于0立刻用一个循环调用ReadEtcInfo, 与后台数据分库同步。

I 读取二次发行结果数据

Ø 函数原型：

```
int _stdcall ReadEtcInfo(byte* ObuSerialNo, byte* status, byte* EtcInfoData, int InfoLen, byte* PicBuf, int* PicLen, byte DelAfterRead);
```

Ø 函数功能：

读取一条发行结果数据，以更新后台数据库记录。

Ø 函数参数：

2 ObuSerialNo, Obu应用序列号，从SyncEtcInfo函数获得。

2 status, 发行状态，0, 发行成功，1, 未发行，2, 发行失败。

2 EtcInfoData, 发行的实际数据内容，由于二次发行可以脱机发行，发行过程中可能人为修改部分参数，因此该项数据分可能与PushEtcInfo函数输入的内容不同。调用者应为该参数分配足够长度的缓冲区。

2 InfoLen, 读出EtcInfoData的长度。

2 PicBuf, 接收文图象件缓冲区，调用者为调用分配足够大的缓冲区。

2 PicLen, 出口参数，为接收到的图象数据的长度。

2 DelAfterRead, 读取后是否删除该记录数据，0, 不删除，1, 删除。

Ø 函数返回：

0, 成功。1, 读取失败，2, 未找到ObuSerialNo相关数据。

Ø 说明：

如果设备不支持图象抓拍功能，PicBuf可以传入NULL, PicLen返回0。