生产线设计文档

**HDIC**

**[键入公司地址]**

**[键入电话号码]**

**[键入传真号码]**

**[选取日期]**

薛长城

本文档是关于生产工具的设计文档，文档依据生产流水线的流程从头到尾较详细的叙述了每一个环节的设计方案。

## 历史修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 版本号 | 说明 | 作者 | 发布日期 |
| 1 | V1.0 | 直播星生产线设计文档 | 薛长城 | 2013.01.13 |

目录

[历史修改记录 1](#_Toc345862305)

[1 Overview 5](#_Toc345862306)

# Terminology

|  |  |
| --- | --- |
| Term | Definition |
| 上位机 | pc端的应用程序 |
| 下位机 | 机顶盒程序 |
| STB | 机顶盒的简称 |
| DB | 服务器上的数据库 |
| STBID | 用户管理中心为每台机顶盒定义的16位十进制数字的唯一编号 |
| Chip ID | 芯片厂家为其安全芯片定义的4字节长的唯一编号 |
| CA ID | 基于芯片序列号生成的11位十进制数字的唯一编号 |
| Smart Card ID | 在村村通中，由用户管理中心提供的智能卡卡号 |
| STBPlatformID | 由用户管理中心定义的含有机顶盒制造商标识、机顶盒型号标识和硬件版本标识的3字节唯一标识编号 |

# Overview(概述)

**1.1序列化流程**

序列化主要包括以下步骤：

1. 创建机顶盒序列号；
2. 计算加密序列号；
3. 将序列化数据写至机顶盒Flash中；
4. 启动机顶盒安全保护；
5. 生成回传给用户管理中心的返回文件；
6. 进行机顶盒安全验证。
7. 下图描述了序列化的整体流程：
8. 

(注意：目前分为两个工位，工位一负责生成序列号数据存入数据库、向Flash写入序列化数据，Flash写保护、打印标签等。工位二负责校验序列化数据、校验加密序列号、校验机顶盒序列号、校验Flash写保护状态、校验高级安全状态等)

# 流程图

## 工位一流程图

按下回车/空格键开始

判断是否重复序列化

PC与STB握手成功，PC向

STB发送获取STBTYPE命令

PC向STB发送获取

CHIPID命令

握手成功

握手失败

生成CAID

将序列化数据转换成88字节数据向STB发送

不重复

重复

PC接收STB发送

握手命令

搜索序列化数据

检索ChipID

显示

不存在

存在

发送失败

由STBType判断机顶盒类型

发送成功

获取失败

获取失败

PC向STB发送获取制造商ID命令（MFID）

获取失败

PC向STB发送获取机顶盒型号命令（MDID）

获取失败

村村通

户户通

显示

显示

显示

详细握手原理请参考“握手协议”

CAID生成过程请参考“CAID生成”模块

PC向STB发送获取硬件ID命令（HDID）

获取失败

生成STBID

PC向STB发送STBID

发送失败

存入数据库

打印条形码进行下一台

开启扫描枪

12位数字

存入数据库

打印条形码进行下一台

扫描枪等待扫描

是

否

显示

显示

显示

STBID生成过程请参考“STBID生成”模块

## 工位二流程图

按下回车/空格键开始

PC与STB握手成功，PC向

STB发送获取STBTYPE命令

PC向STB发送获取

CHIPID命令

握手成功

PC接收STB发送

握手命令

检索ChipID

握手失败

获取失败

获取失败

判断序列化表中是否序列化

序列化

判断是否已经校验过

没有序列化

是

PC向STB发送获取

Flash写保护状态

获取失败

判断Flash写保护状态

未写保护

当前状态：写保护

否

PC向STB发送获取

高级安全状态

判断高级安全状态

获取失败

高级安全未打开

PC向STB发送获取

序列化数据

获取失败

与数据库中的序列化数据进行比较

数据不一致

开启扫描仪

11位数字

12位数字

16位数字

PC向机顶盒发送获取

CAID命令

PC向机顶盒发送获取

STBID命令

该CAID与标签比对

标签CAID与数据库比对

数据不一致

标签智能卡号与数据库比对

该STBID与标签比对

标签STBID与数据库比对

校验成功，修改校验标识，进行下一台

不一致

# 协议

## 握手协议

Start

**STB**

Send

COM\_CONNECT

**PC**

Get

COM\_CONNECT

Get

COM\_OK

Get

COM\_CONNECT

Get

COM\_ASKHAND

**Y**

**Y**

Send

COM\_OK

Send

COM\_OK

握手成功，开始通信

## 通信格式

通信方式使用串口 (UART)

表1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Bits** | **Parity** | **Stop Bit** | **Baud Rate** | **Flow Control** |
| 8 | None | 1 | 115200 bps | N |

* 机顶盒启动后，下位机轮询发送握手信号（使用了超时机制）
* pc端启动后，上位机轮询接收握手信号（使用了超时机制1分钟）
  1. **数据格式**

表2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Start  Code** | **Command  one** | **Command  two** | **Command  Length** | **Command  Data** | **CHECKSUM** |
| 1 Byte | 1 Byte | 1 Byte | 1 Bytes | n Bytes | * 1. Byte |

* Start Code: 命令头，固定为0x48
* Command one: 发送的命令
* Command two: 备用
* Command Length:发送数据的长度（仅仅是数据段的长度）
* Command Data: 发送的数据
* CHECKSUM：数据包校验（所有数据相加，忽略溢出的数据）

|  |  |  |
| --- | --- | --- |
| **命令**  **Command Code** | **描述**  **Description** | **示例**  **Example** |
| COM\_ASKHAND | 下位机发送握手命令 | [0x48][COM\_ASKHAND][ COM\_RETURN  ][DATA\_LENGTH][CHECKSUM] |
| COM\_CONNECT | 上位机握手后回复命令 | 上位机回复：  [0x48][ COM\_CONNECT][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_HANDINFO | 下位机发送确认命令 | [0x48][ COM\_HANDINFO][ COM\_RETURN  ][DATA\_LENGTH][CHECKSUM] |
| COM\_OK | 上位机回复确认命令 | [0x48][ COM\_OK][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_START | 下位机发送开始接收数据命令 | [0x48][ COM\_START][ COM\_RETURN  ][DATA\_LENGTH][CHECKSUM] |
| COM\_STBTYPE | 上位机发送获取STBType命令 | [0x48][ COM\_STBTYPE][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_STBTYPE | 下位机回复STBType命令 | [0x48][ COM\_STBTYPE][ COM\_RETURN  ][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_CHIPID | 上位机发送获取ChipID命令 | [0x48][ COM\_CHIPID][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_CHIPID | 下位机回复ChipID命令 | [0x48][ COM\_CHIPID][ COM\_RETURN  ][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_MFID | 上位机发送获取制造商ID命令 | [0x48][ COM\_MFID][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_MFID | 下位机回复制造商ID命令 | [0x48][ COM\_MFID][ COM\_RETURN  ][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_MDID | 上位机发送获取机顶盒型号命令 | [0x48][ COM\_MDID][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_MDID | 下位机回复机顶盒型号命令 | [0x48][ COM\_MDID][ COM\_RETURN  ][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_HWID | 上位机发送获取硬件ID命令 | [0x48][ COM\_HWID][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_HWID | 下位机回复机硬件ID命令 | [0x48][ COM\_HWID][ COM\_RETURN  ][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_STBIDPC | 上位机发送STBID命令 | [0x48][ COM\_STBIDPC][ COM\_NULL  ][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_STBIDPCOK | 下位机回复收到STBID命令 | [0x48][ COM\_STBIDPCOK][ COM\_RETURN][DATA\_LENGTH][CHECKSUM] |
| COM\_FLASHWRITELICENSE | 上位机发送序列化数据命令 | [0x48][ COM\_FLASHWRITELICENSE][ COM\_NULL][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_FLASHWRITELICENSEOK | 下位机回复收到序列化数据命令 | [0x48][ COM\_FLASHWRITELICENSEOK][ COM\_RETURN||COM\_FAIL][DATA\_LENGTH][CHECKSUM] |
| COM\_FLASHSTATUS | 上位机发送获取Flash写保护状态命令 | [0x48][ COM\_FLASHSTATUS][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_FLASHSTATUS | 下位机回复Flash当前状态命令 | [0x48][ COM\_FLASHSTATUS][ COM\_RETURN  ][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_SECURITYSTATUS | 上位机发送获取高级安全状态命令 | [0x48][ COM\_SECURITYSTATUS][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_SECURITYSTATUS | 下位机回复高级安全当前状态 | [0x48][ COM\_SECURITYSTATUS][ COM\_RETURN  ][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_GETLICENSE | 上位机获取序列化数据命令 | [0x48][ COM\_GETLICENSE][ COM\_NULL  ][DATA\_LENGTH][CHECKSUM] |
| COM\_GETLICENSEOK | 下位机回复序列化数据命令 | [0x48][ COM\_GETLICENSEOK][ COM\_RETURN||COM\_FAIL][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_CAID | 上位机发送获取CAID命令 | [0x48][ COM\_CAID][ COM\_NULL][DATA\_LENGTH][CHECKSUM] |
| COM\_CAID | 下位机回复CAID命令 | [0x48][ COM\_CAID][ COM\_RETURN][DATA\_LENGTH][Data][CHECKSUM] |
| COM\_STBIDSTB | 上位机发送获取STBID命令 | [0x48][ COM\_STBIDSTB][ COM\_NULL][DATA\_LENGTH][CHECKSUM] |
| COM\_STBIDSTB | 下位机回复STBID命令 | [0x48][ COM\_STBIDSTB][ COM\_RETURN][DATA\_LENGTH][Data][CHECKSUM] |

# 序列化数据

## CAID算法

CAID（加密序列号）是基于芯片序列号生成的11位十进制数字的唯一编号。具体算法如下：

CA ID = QQQQQQQQQQZ, 其中

QQQQQQQQQQ(十进制) = 芯片序列号 ^ 0x80000000

z=QQQQQQQQQQ的校验位，基于Luhn算法得出，Luhn算法实现过程如下：

private bool CalculateLuhnAlgorithm(int[] digits, int length)

{

int i;

int sum = 0;

bool alt = true;

for (i = length - 2; i >= 0; i--)

{

if (alt)

{

int temp = digits[i] \* 2;

if (temp > 9)

{

temp -= 9;

}

sum += temp;

}

else

{

sum += digits[i];

}

alt = !alt;

}

int modulo = sum % 10;

if (modulo > 0)

{

digits[length - 1] = ((10 - modulo));

}

else

{

digits[length - 1] = 0;

}

return true;

}

## STBID生成

长度为48bit“STB ID”（加8bit预留位），采用16位十进制数字表示，从前向后的顺序依次为：3位的制造商标识（8bit），2位的机顶盒类型号（6bit），２位生产年号（6bit）、２位生产周号（6bit）、7位生产流水号（22bit），流水号为全年所有批次、型号的累计流水号，若流水号不足7位，在前面补零。

用二进制表示的48位机顶盒序列号为：

Manufacturer\_ID / STB type / Production\_Year / Production\_Week / Production\_Serial\_Number

制造商ID / 机顶盒类型号 / 生产年号 / 生产周号 / 生产流水号

01111000/ 000000 / 001001 / 100100 / 1000010110101010110000

//第一.获取今天是一年当中的第几天

int iDays = DateTime.Today.DayOfYear;

//第二.找到第一周的最后一天（先获取1月1日是周几，从而得知第一周周末是周几）

int firstWeekend = 7 - Convert.ToInt32(DateTime.Parse(nowdate.Year + "-1-1").DayOfWeek);

//第三.（今天 减去 第一周周末）/7 等于 距第一周有多少周 再加上第一周的1 就是今天是今年的第几周了 // 刚好考虑了惟一的特殊情况就是，今天刚好在第一周内，那么距第一周就是0 再加上第一周的1 最后还是1

int weeknums = Convert.ToInt32(Math.Ceiling((iDays - firstWeekend) / 7.0)) + 1;