# SAM4A 非接触 IC 卡大读卡机使用说明书

版本 1.6.42

2015 年 12 月 23 日

苏州市永兴电子有限公司

# 目录

# 1. 产品概要

- 本读卡机基于 ARM Cortex A 系列处理器，最大时钟频率达到 1GHz，并且拥有大量的外部 Flash 和 SDRAM 扩展，DDR3 RAM 内存为 1G 字节，NAND Flash 标准尺寸为 8G 字节,最大支持 32G 字节. 非常适合使用嵌入式操作系统完成复杂的功能. 该处理器内部带有 AES,SHA 和三重 DES 加密硬件，非常适合 CPU 卡的加密操作.

- 读卡机默认使用 Linux 操作系统，内核版本为 3.14，用户可以将读卡机替换掉原闸机内工控机. 厂商将提供关于射频芯片，SAM 卡，GPIO 等底层驱动及 API 接口，以方便用户直接调用. 用户还可以自行修改操作系统，改为 WindowsCE 或是 Android.

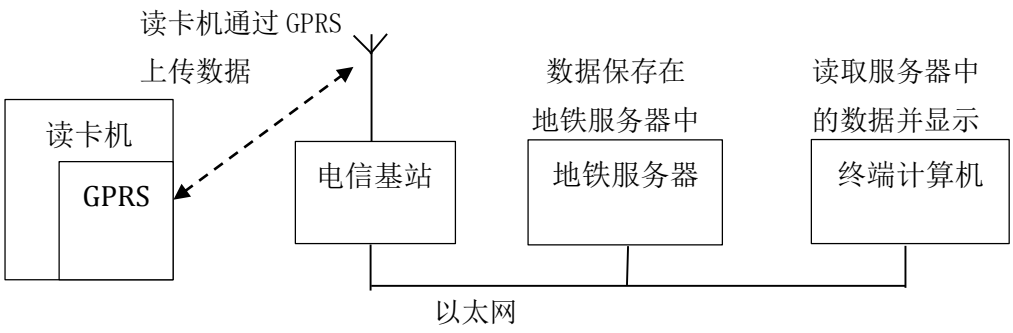
- 本读卡机可以通过 HDMI 接口连接到 LCD 液晶显示器，以提供视频和音频. 用户可以选配 HDMI 转 VGA 接口，并通过 VGA 连接到显示器上. 读卡机视频提供 16 位 RGB 色深,分辨率为 640*480 像素,最大为 1920*1080 像素. 音频则为双声道立体声输出.

- 本读卡机拥有两个独立 HF 射频通道，可对满足 ISO14443 Type A/ TypeB 协议的非接触 IC 卡和上海模式卡进行读写等操作，通道 1 同时还支持 SONY Felica（TYPE C）系列卡的读写操作. 读卡机同时还有 1 个 UHF 射频通道，支持 Zigbee 无线通信功能.

- 该读卡机具有较强的防电磁能力,手机及地铁等电子电气设备的常规使用不会对 IC 卡标准读写模块的使用造成影响. 本读卡机已通过国家电磁兼容性鉴定,无线电骚扰限值符合 GB9254 中相应级别 B 级的规定，抗扰度限值符合 GB/T17618 中的相应规定.

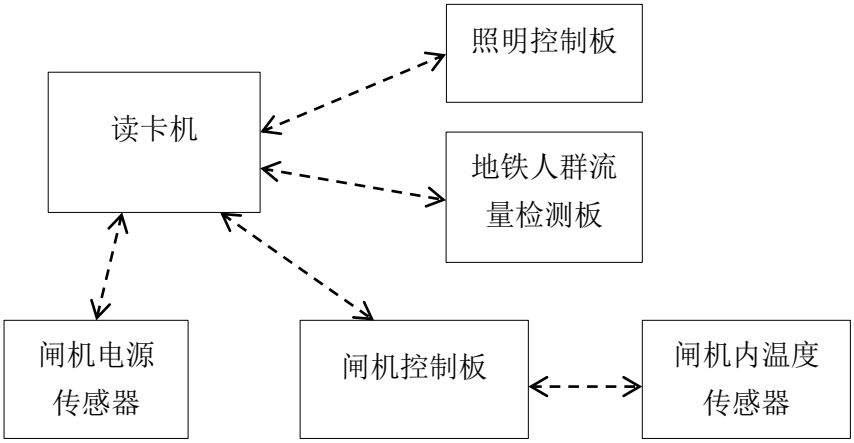
- 通信接口方面，本读卡机有 3 个串口，支持 RS232/ RS485/ CAN 三种通信标准. 有一个 RJ45 接口，可以连接到以太网. 两个 USB A 类插座，可以作为主机连接到诸如 U 盘的外设，也可作为客户端连接到 PC 上. 一个 SD 卡插座，支持 MMC4.3 和 SD2.0 标准，可以插入 MMC 卡或是 SD 卡，并支持从 SD 卡启动操作系统.

- 本读卡机带有 Zigbee 无线局域网或 GPRS 分组无线移动数据业务功能. 使用 GPRS 能让读卡机实时上传刷卡数据，完成考勤,巡更等功能. 下图显示了一个巡更系统的例子，巡更人员的刷卡信息通过 GPRS 上传至基站，再保存到地铁服务器中. 中心的终端计算机读取服务器中的数据，并显示出来.

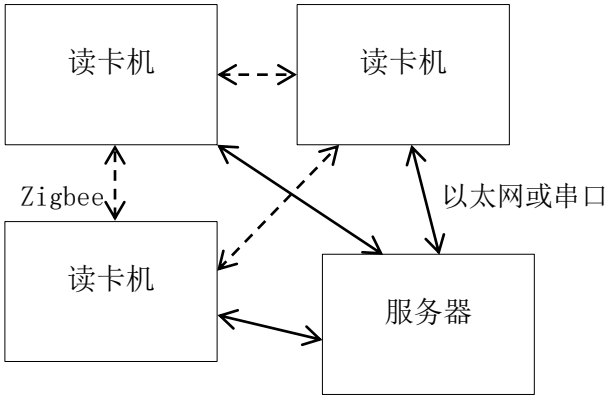


- Zigbee 无线局域网可以连接到其他的 Zigbee 设备，如 Zigbee 传感器或其他读卡机，从而构成一个 Zigbee 设备网络．下面简单列举几个 Zigbee 网络的用例．

  - Zigbee 传感器网络．读卡机可以通过 Zigbee 无线网络连接到其他传感器上，构成如下图的网络．当闸机电源传感器发现电网供给的 220V 浪涌较大，输出电压不稳定时，通过网络将信息发送给读卡机，读卡机执行掉电保护操作，以防止突然掉电带来的数据丢失．地铁人群流量检测板则不断将当前流量发送给读卡机，当读卡人数较少时，读卡机进入省电模式，降低寻卡频率．与此同时，读卡机也将人数较少的信息发送给照明控制板， 照明控制板将该站的照明亮度降低，以节约更多电量．




  - 读卡机网络．多个读卡机之间也可以建立 Zigbee 网络，这改变了读卡机与服务器之间的连接方式，使得读卡机之间可以相互通信．例如，如果需要校验黑名单的正确性，不再需要从服务器下载另一份数据．而是与另一台读卡机通信，相互比对黑名单文件校验值即可．



# 2. 读卡机规格

## 2.1 读卡机参数

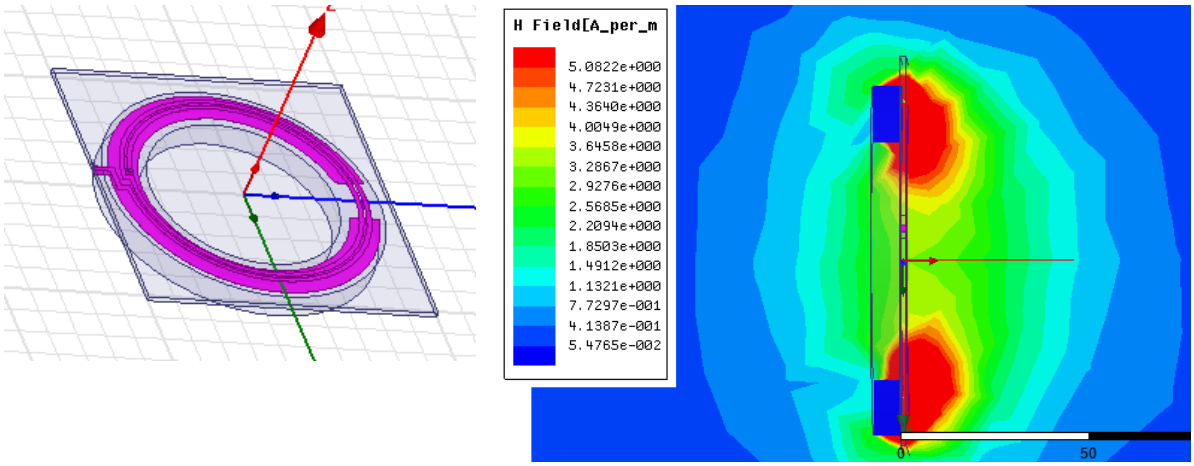
| 读卡机硬件参数 | |
|---|---|
| 读卡机处理器 | AM3352 ARM Cortex A8 系列内核 |
| 处理器最大频率 | 1GHz |
| 处理器内部 ROM | 176Kbytes |
| 处理器内部 RAM | 128Kbytes |
| 扩展并行 Flash | 并行 NAND Flash,标准尺寸为 8G 字节, 最大可扩展至 32G 字节. |
| 扩展 RAM | 采用 DDR3 SDRAM, 标准尺寸为 1G 字节. |
| 扩展 EEROM | 采用铁电存储器 FRAM, 标准尺寸为 128K 字节 |
| 处理器加密硬件 | 有(支持 AES, SHA, PKA, RNG 算法) |
| 无线局域网络 | Zigbee: 支持 IEEE802.14.5 标准 GPRS: 支持 GSM2/2+标准 |
| RS232 / RS485 / CAN 串口 | 三个 DB9 插座, 插座 J9 为通信串口,支持 RS232 或 RS485.<br>插座 J11 支持 RS232,并支持 2 个 GPIO 脚. 插座 J10 支持 RS232 或 CAN 总线. |
| 实时时钟功能(RTC) | 有,需要添加纽扣电池 |
| USB | 2 个(两个 USB A 型插座,一个为 USB2.0 高速主机.另一个 USB2.0 高速设备) |
| SAM 卡插座 | 8 个 SAM 卡. 支持 ISO/IEC7816-1/2 标准 PPS 协议,<br>可支持 9600—115200 波特率的 SAM 卡 |
| 射频通道 | 3 个(2 个 HF 射频通道,13.56MHz. 1 个 UHF 射频通道,2.4GHz) |
| SD 卡接口 | 有(标准 SD 卡插座, 支持 MMC 卡 4.3 标准和 SD 存储卡 2.0 标准) |
| 以太网接口 | 有(RJ45 插座, 10/100M 以太网) |
| HDMI 接口 | 有(16 位 RGB 色深,分辨率最大为 1920*1080 像素.音频为双声道立体声输出) |
| LED 状态显示 | 4 个 LED,一个电源指示灯,三个用户可编程状态指示灯,<br>1 个用户可编程 4 位 8 段数码管 |
| 按钮 | 2 个, 一个复位按钮, 另一个为用户可编程按钮 |
| 蜂鸣器 | 有 |
| **读卡机软件参数** | |
| 操作系统支持 | 支持 Linux(默认), Android, WinCE 等操作系统 |
| UBOOT 版本 | 2014.07 |
| Linux 内核版本 | 3.14 |
| 支持文件系统 | FAT/Ext2/Ext3/JFFS2/Ubifs |
| 典型文件系统尺寸<br>(除去引导程序和操作系统尺寸) | 7.5G 字节<br>(按照北京地铁标准计算, 一卡通黑名单每条 8 字节, 一卡通检票交易每条 106 字节. 最大支持超过 2 千万条黑名单和 1 千万条检票交易) |

| | |
|---|---|
| **通信参数** | |
| 以太网通信速率 | 10/100Mbps |
| USB 标准 | USB2.0 High Speed |
| USB 数据速率 | 480Mbps，兼容 12Mb/s 或 1.5Mb/s |
| 通信串口波特率 | 默认为 115200bps，用户可自定义 |
| 通信串口参数 | 8 位数据位,1 位停止位,无奇偶校验 |
| SD 传输速率 | 支持 4 位宽总线，默认速度模式最大数据传输速率为 12.5Mb/s<br>高速模式下最大数据传输速率为 25Mb/s |
| HDMI 传输速率 | 分辨率为 640*480 像素时，像素时钟 25.175MHz,<br>TMDS 单通道写穿速率 251Mb/s |
| **射频参数** | |
| 射频协议 ISO14443,ISO/IEC18000-Part3，ISO/IEC 15693 及 JIS X 6319-4 | |
| 载波频率 | 13.56MHz |
| 通讯速率 | 106 Kbps 可支持 212Kbps,424Kbps,848Kbps |
| 调制模式 | OOK |
| IC 卡标准 | 支持 NXP 的 Mifare Classic 系列（M1S50,M1S70,Ultralight）非接触 IC 卡<br>支持符合 ISO/IEC14443 TYPE A 标准的上海模式 Mifare 系列非接触 IC 卡<br>支持符合 ISO/IEC14443 TYPE A 标准的 CPU 卡<br>支持符合 ISO/IEC14443 TYPE B 标准的非接触 IC 卡<br>通道 1 支持符合 JIS X 6319-4 标准的 SONY Felica 非接触 IC 卡 |
| HF 天线参数 | |
| 天线类型 | Loop |
| 近场磁场强度 | 天线表面磁场强度≤7.5A/m rms，5cm 处磁场强度≥1.5A/m rms |
| 特征阻抗 | 50 Ohm |
| 射频协议　IEEE 802.15.4/Zigbee | |
| 载波频率 | 2.4GHz |
| 通讯速率 | 250Kbps |
| 调制模式 | O-QPSK |
| 频谱扩散 | DSSS |
| 功放输出 | 0dBm 至 5dBm 可调 |
| 接收机灵敏度 | -98dBm |
| UHF 天线参数 | |
| 天线类型 | PIFA |
| 半功率束宽(HPBW) | 86° 当 Phi=0 度 |
| 一零束宽(FNBW) | 145° 当 Phi=0 度 |
| 向性 | 4.19dB |
| 天线阻抗 | 31.9-46.5j Ohm@2.4GHz |

| | |
|---|---|
| **电磁兼容性参数** | |
| 辐射和传导标准 | 无线电骚扰限值符合 GB9254 中相应级别 B 级的规定<br>GB/T9254-1998（EN 55022） |
| 静电放电 | GB/T17626.2-1998（IEC 61000-4-2） |
| 脉冲群 | GB/T17626.4-1998（IEC 61000-4-4） |
| 浪涌抗扰度 | GB/T17626.5-1998（IEC 61000-4-5） |
| 电压暂降，暂时中断和电压变化抗扰度 | GB/T17626.11-1998（IEC 61000-4-11） |
| **机械参数** | |
| 读卡机主板尺寸 | 长 141mm*宽 96mm*高 15mm |
| 读卡机主板定位孔 | 定位孔直径 3.5mm，使用标准 M3 平头螺丝安装. |
| **电气参数** | |
| 工作温度 | -20℃~70℃ |
| 工作湿度 | ≤90% |
| 电源电压 | DC12V±10% |
| 静态电流 | 220mA@12.0V |
| 绝对最大电压 | DC20V |
| 最大功率损耗 | 4W |

## 2.2 HF 天线参数

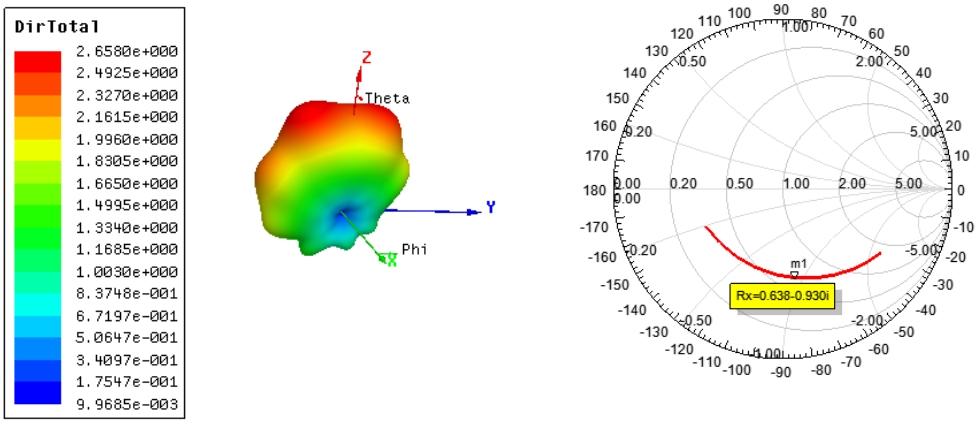
本读卡机使用近场环状天线，经过阻抗匹配网络匹配至 50 欧姆. 按照 ISO14443 标准，天线表面磁场强度应小于 7.5A/m 均方根，5cm 处磁场强度应大于 1.5A/m 均方根. 仿真结果显示本产品天线表面磁场强度为 5.08A/m，5cm 处磁场强度为 1.85A/m，符合标准要求.



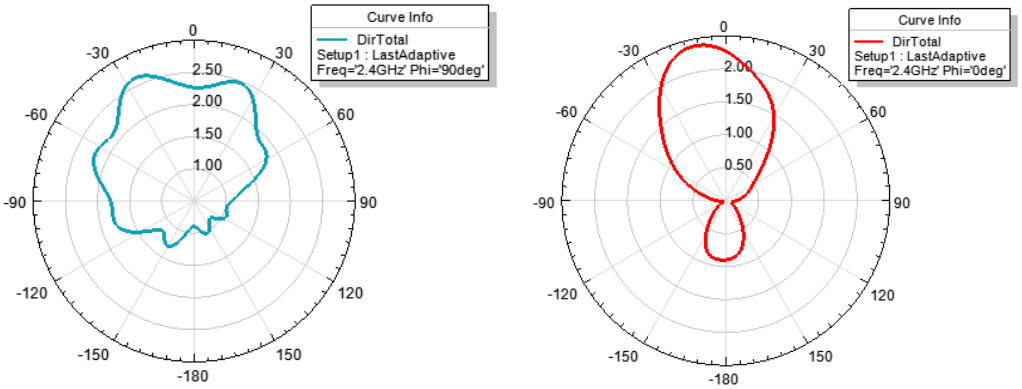

对于标准 Mifare 卡，本产品典型读卡距离为 8cm，最大读卡距离为 10cm.

## 2.3 UHF 天线参数

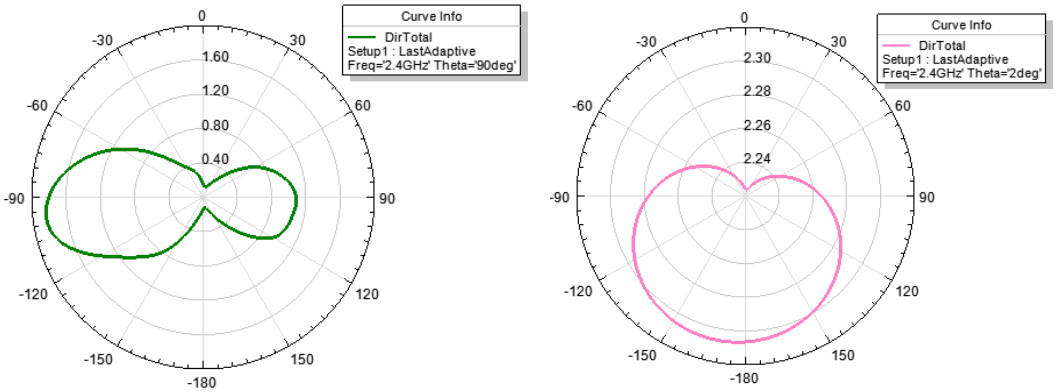
当读卡机有 Zigbee 功能时，通常用户可以选购一个 2.4GHz 的偶极子天线连接到 SMA 插座 J20 上. 与此同时，读卡机有一个 PCB 天线，当读卡机为塑料机壳时，可以使用该天线. 在下面描述中,我们定义 SMA 插座 J20 指向的反方向为 Y 轴，则天线辐射特征和 $S_{11}$ 参数如下图所示. 在 2.4GHz 上，$S_{11}$=0.077-0.524j $Z_{11}$=31.9-46.5jΩ.



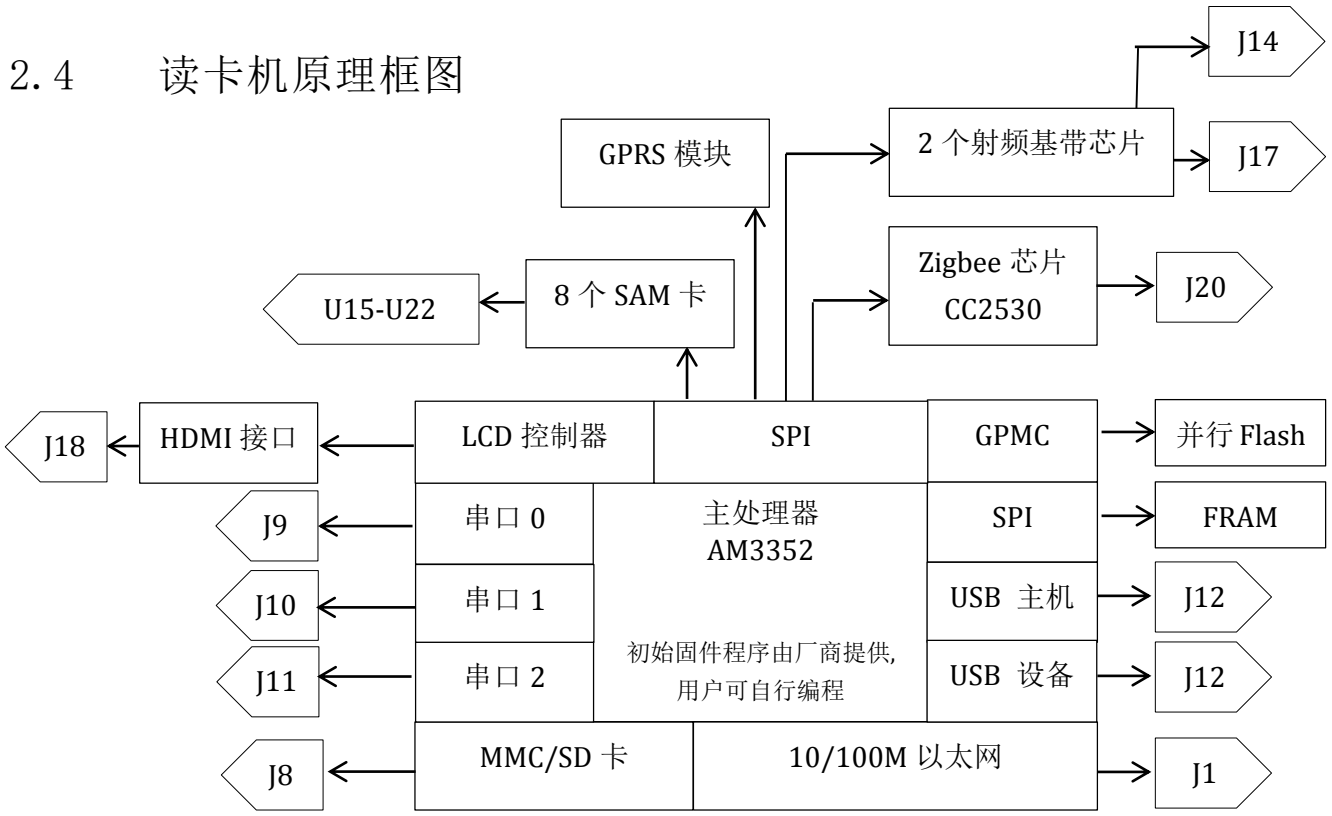

当 φ 为 90 度和 0 度时，辐射特征如下图所示,YZ 平面天线向性为 4.19dB.



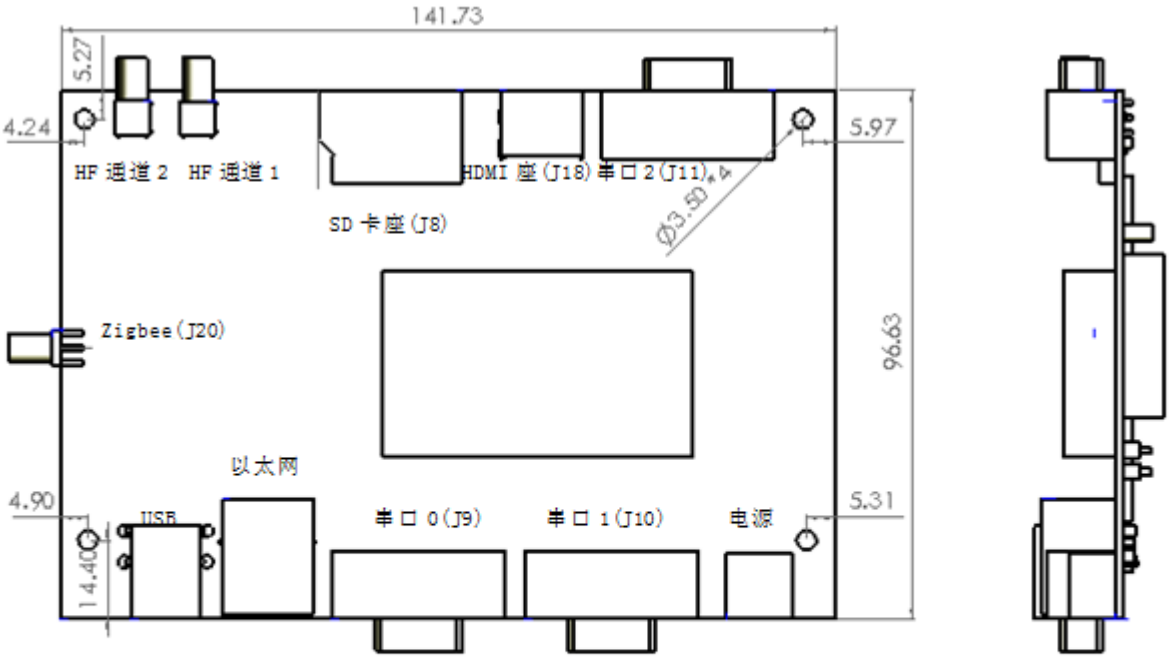

当 θ 为 90 度和 0 度时，辐射特征如下图所示,XZ 平面天线向性为 3.63dB，XY 平面天线向性为 2.65dB.

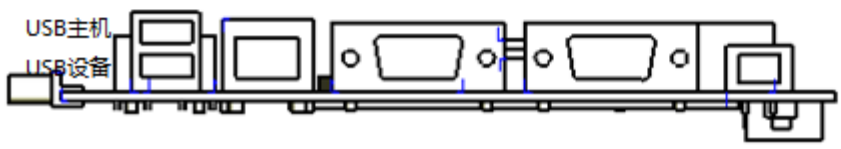


## 2.4 读卡机原理框图




## 2.5 读卡机电路板结构



## 2.6 读卡机机械结构

## 2.7 读卡机订购信息

SAM4A 读卡机 HF 调制芯片默认使用 RC531,支持标准 Mifare 卡. 如果用户需要, HF 调制芯片也可以使用 FM1715 并支持上海秘钥集,能够读取上海 Mifare 卡, 如基于 FM11RF08SH 的非接触卡.

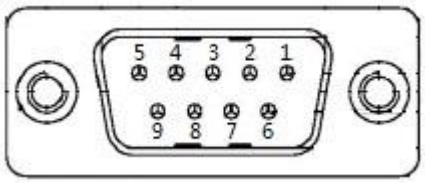
| 读卡机型号 | SAM4A-A (基本型) | SAM4A-B (ZigBee 型) | SAM4A-C (GPRS 型) |
|---|---|---|---|
| HF 射频通道 1 | RC531 或 FM1715 (ISO14443 TypeA/ TypeB) | RC663 (ISO14443 TypeA/ TypeB/ SONY Felica) | RC663 (ISO14443 TypeA/ TypeB/ SONY Felica) |
| HF 射频通道 2 | RC531 或 FM1715 (ISO14443 TypeA/ TypeB) | RC531 或 FM1715 (ISO14443 TypeA/ TypeB) | RC531 或 FM1715 (ISO14443 TypeA/ TypeB) |
| UHF 射频通道 | 无 | 有, Zigbee | 有, GPRS |
| HDMI 接口 | 无 | 有 | 有 |
| Flash | 8G 字节 | 8G 字节,最大至 32G 字节 | 8G 字节,最大至 32G 字节 |
| DDR3 SDRAM | 1G 字节 | 1G 字节 | 1G 字节 |
| FRAM | 128K 字节 | 128K 字节 | 128K 字节 |

# 3. 系统接口定义

本章定义了读卡机系统的硬件接口,包括插座引脚定义等.

## 3.1 串口定义

读卡机的串口使用标准 DB9 插座，可以很方便地和计算机串口连接. 读卡机有三个 Female(孔)串口插座，关于串口的位置，参考 2.5 节.关于串口的软件定义，参考 4.2.7 节. 串口 0 用于和上位机通信，插座编号为 J9，引脚 9 通过一个 0.4A 自恢复保险丝接到 5V，用于对外提供电源. 如有需要，该串口还兼容 RS422/485 标准. 默认情况下带有一个 120 欧姆终端负载电阻，该终端电阻可以根据用户需要改变. 具体引脚定义如下:




| 串口 0 信号 | 引脚 | 引脚说明 | 方向 |
|---|---|---|---|
| TXD | 2 | 读卡机→上位机 | OUT |
| RXD | 3 | 读卡机←上位机 | IN |
| GND | 5 | 地线 | N/A |
| VCC | 9 | 5V 电源 | N/A |
| RS485_B(可选) | 1 | RS485 标准差分数据线 | N/A |
| RS485_A(可选) | 8 | | N/A |

串口 1 用于和上位机通信，插座编号为 J10，引脚 9 通过一个 0.4A 自恢复保险丝接到 5V，用于对外提供电源. 如有需要，该串口还兼容 CAN 总线. CAN 总线默认情况下带有一个 120 欧姆终端负载电阻.

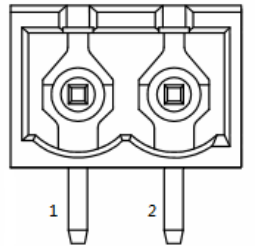
| 串口 1 信号 | 引脚 | 引脚说明 | 方向 |
|---|---|---|---|
| TXD | 2 | 读卡机→上位机 | OUT |
| RXD | 3 | 读卡机←上位机 | IN |
| GND | 5 | 地线 | N/A |
| VCC | 9 | 5V 电源 | N/A |
| CANH(可选) | 1 | CAN 总线 | N/A |
| CANL(可选) | 8 | | N/A |

串口 2 既可以用于和上位机通信，也可以与手机模块进行通信．插座编号为 J11，引脚 9 通过一个 0.4A 自恢复保险丝接到 5V，用于对外提供电源.引脚 1，6 为 CMOS 逻辑 GPIO 引脚，输入输出方向用户可以自定义，也可以用于从读卡机到手机模块的控制线.

| 串口 2 信号 | 引脚 | 引脚说明 | 方向 |
|---|---|---|---|
| TXD | 2 | 读卡机→上位机 | OUT |
| RXD | 3 | 读卡机←上位机 | IN |
| GPIO1 | 1 | 读卡机→手机模块 | INOUT |
| GPIO2 | 6 | 读卡机→手机模块 | INOUT |
| GND | 5 | 地线 | N/A |
| VCC | 9 | 5V 电源 | N/A |

## 3.2 电源插座定义

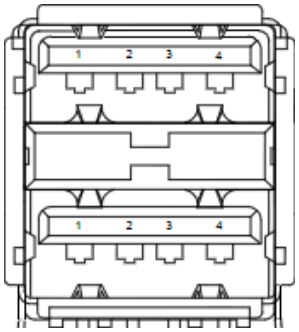
读卡机的电压插座使用弯针 HT5.08 座，插座编号为 J6，插入时请注意方向和缺口．读卡机内部带有反接保护电路，在电源和地接反时，读卡机并不会损坏．但是，请注意输入电源电压，最大不能超过 20V．读卡机最低能在大约 8V 时工作，过低的电源电源会影响读卡机的正常工作.




| 电源座引脚 | 定义 | 说明 |
|---|---|---|
| 1 | DC12V | 12V 正电源 |
| 2 | GND | 地 |

## 3.3 USB 插座定义

本读卡机有两个的 USB 插座．插座编号为 J12 的双层 USB A 型插座连接到主处理器.其下面的插座为 USB2.0 全速主机，电源能够提供 0.4A 电流．上面的插座为 USB2.0 全速设备，用于和上位机连接(读卡机 SAM 卡一面朝上)．具体信息请参考 USB2.0 标准第 6 章.



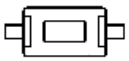
| 双排 A 座引脚 | 定义 | 说明 |
|---|---|---|
| 上层 1 | 设备 VBUS | 必须由主机提供一个 5V 电源，消耗电流不大于 10mA |
| 上层 2 | 设备 D- | 负数据线 |
| 上层 3 | 设备 D+ | 正数据线 |
| 上层 4 | 设备 GND | 地 |
| 下层 1 | 主机 VBUS | USB 5V 电源，对外最大提供 0.4A 电流 |
| 下层 2 | 主机 D- | 负数据线 |
| 下层 3 | 主机 D+ | 正数据线 |
| 下层 4 | 主机 GND | 地 |

## 3.4 射频插座定义

读卡机的 HF 射频插座使用 SMA 90 度弯角母座(下图左侧)，特征阻抗 50 欧姆. 插座编号分别为 J14 和 J17. J14 对应射频通道 1，J17 对应射频通道 2. UHF 射频插座使用 SMA 板边缘母座(下图右侧)，特征阻抗 50 欧姆，插座编号 J20.

## 3.5 按钮定义

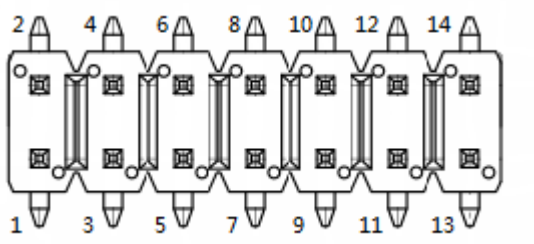
SAM4A 读卡机有两个微动开关，J3 和 J21. 其中，J3 为系统复位按钮，按下后将强迫系统进行软复位. J21 为用户自定义按钮，在系统启动时，引导程序会检查该按钮是否被按下，如果按下，则会擦除 Flash 上所有程序，包括引导程序本身. 关于引导程序，请参考《SAM4A 编程手册》 第 2.1 节.



| 按钮编号 | 定义 | 说明 |
|---|---|---|
| J3 | nRESET | 系统软复位 |
| J21 | BUTTON | 用户自定义(在复位时作为擦除按钮) |

## 3.6 JTAG 插座定义

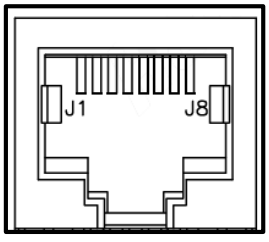
JTAG 插座编号为 J13，是 DC3 2.54mm 间距，14pin 双排直插插针插座. 其引脚定义如下，详细描述请参考 IEEE 1149.1 边界扫描信号标准:




| JTAG 插座引脚 | 定义 | 说明 |
|---|---|---|
| 1 | TMS | JTAG 的 TMS 引脚 |
| 2 | nTRST | JTAG 的复位引脚 |
| 3 | TDI | JTAG 的 TDI 引脚 |
| 4 | GND | 数字地 |
| 5 | VDD3.3 | 3.3V 电源,无过流保护 |
| 6 | NC | 无连接 |
| 7 | TDO | JTAG 的 TDO 引脚 |
| 8 | GND | 数字地 |
| 9 | RTCK | JTAG 的 RTCK 引脚 |
| 10 | GND | 数字地 |
| 11 | TCK | JTAG 的 TCK 引脚 |
| 12 | GND | 数字地 |
| 13 | EMU0 | 调试仿真引脚 0 |
| 14 | EMU1 | 调试仿真引脚 1 |

## 3.7 以太网插座定义

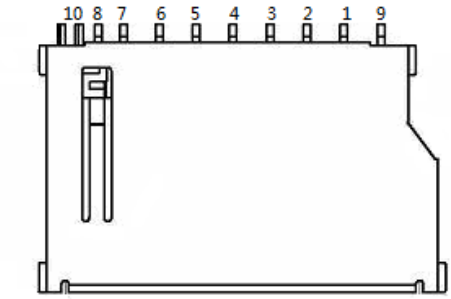
以太网插座使用 RJ45 插座，连接器为 8J8C 类型，详细描述请参考 ISO/IEC 15018 和 ISO/IEC 11801 标准. 插座编号为 J9，其引脚定义如下:



| 以太网插座引脚 | 定义 |
|---|---|
| 1 | TX+ |
| 2 | TX- |
| 3 | RX+ |
| 4 | SHIELD |
| 5 | SHIELD |
| 6 | RX- |
| 7 | SHIELD |
| 8 | SHIELD |

## 3.8 SD 卡插座定义

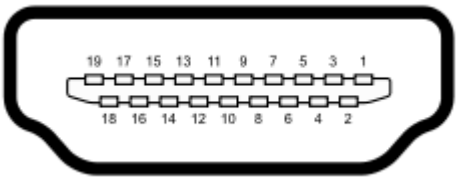
SD 卡插座使用标准 SD 短体卡槽，详细描述请参考 SD Card Association 物理层标准. 插座编号为 J8，其引脚定义如下:




| SD 卡插座引脚 | 定义 |
|---|---|
| 1 | CD/DAT3 |
| 2 | CMD |
| 3 | VSS1 |
| 4 | VDD |
| 5 | CLK |
| 6 | VSS2 |
| 7 | DAT0 |
| 8 | DAT1 |
| 9 | DAT2 |
| 10 | DETECT |

## 3.9 HDMI 插座定义

HDMI 插座使用标准的 Type A HDMI 接头，详细描述请参考 HDMI2.0 标准. 插座编号为 J18，其引脚定义如下:




| HDMI 插座引脚 | 定义 |
|---|---|
| 1 | TMDS Data2+ |
| 2 | TMDS Data2 Shield |
| 3 | TMDS Data2－ |
| 4 | TMDS Data1+ |
| 5 | TMDS Data1 Shield |
| 6 | TMDS Data1－ |
| 7 | TMDS Data0+ |
| 8 | TMDS Data0 Shield |
| 9 | TMDS Data0－ |
| 10 | TMDS Clock+ |
| 11 | TMDS Clock Shield |
| 12 | TMDS Clock－ |
| 13 | CEC |
| 14 | NC |
| 15 | SCL |
| 16 | SDA |
| 17 | DDC/CEC Ground |
| 18 | +5V Power |
| 19 | Hot Plug Detect |

# 4. 系统软件

本章描述 SAM4A 读卡机附带的支持软件，底层驱动和动态链接库均使用 C 编写，符合 C99 标准，用户可以使用任何支持 C99 标准的编译器进行编译. 类型则由 stdint 头文件定义. 用户在调用时,可以参考驱动库手册. 具体的编程调试过程，请参考《SAM4A 编程手册》第 3 章.

SAM4A 读卡机附带的 Linux 驱动软件和 API 驱动动态库使用许可证 Creative Commons Attribution-NoDerivatives 4.0 International License. 用户可以将本代码用于商业用途，但用户如果修改过本代码，则不得发布经过修改后的代码. 应用程序接口的加密部分使用了 OpenSSL 库，其许可证为 OpenSSL 许可证.

## 4.0 驱动库版本

| 当前版本 | 1.05 |
|---|---|
| 功能 | **暂不支持 HDMI 接口/ Zigbee 无线网络 / ISO14443 Type C 卡操作** |
| 版本历史(pre-beta 除外) | |
| 1.00 | 修改了 RSA 加密函数接口<br>将需要链接到 RSA 的驱动库单独编译为 libsam4a_driver_rsa_arm.so |
| 1.01 | 增加了硬件参数函数，可以读取当前硬件版本信息<br>FERAM 改用 FM25V10 芯片，尺寸增加到 128K 字节 |
| 1.02 | 修改了 GPIO 驱动，去除原 sam4a_gpio.ko 驱动，在 Linux 系统下 GPIO 操作将由 GPIOLib 完成. |
| 1.03 | 去除看门狗函数，看门狗的功能将由 Linux 内部看门狗驱动完成 |
| 1.04 | 修改了 SAM 卡时序，使其支持大部分卡<br>修改了 UART 波特率参数，使 SAM 卡支持波特率 56000<br>添加了对 TypeB 卡的支持<br>系统自动挂载 USB 设备 |
| 1.05 | 添加了电源管理函数，硬件随机数生成函数<br>添加了厂商自动测试程序<br>更新了文件系统，现在支持原生 gcc 编译及远程 gdb 调试功能<br>更新了文件系统使用的库. glibc 库版本从 2.15 升级到 2.20 版本 |

**注意：** 请参考编程手册 4.3.4 节更新系统，选择并下载需要的文件系统.
从 1.05 版本开始，由于更新了库版本，用户需要严格按照编程手册 3.4 和 3.5 节执行，替换原有工具链的库文件并重新编译程序. 否则可能因库版本不匹配，导致程序无法运行或调试.

# 4.1 底层驱动

在提供给用户的 Linux 操作系统中，在系统完成引导后，将自动加载内核驱动模块，用户在应用程序中可以调用这些驱动模块完成指定功能. 对内核操作的流程大致为:

1) 调用 open 函数打开设备驱动.

```
nSPIDev=open ("/dev/sam4a_spi",O_RDWR);
```

2) 调用 read 和 write 函数对设备进行操作.

```
read(nSPIDev,&data,4);
```

3) 调用 close 函数关闭设备.

```
close(nSPIDev);
```

上述几个函数都由 POSIX 定义，具体使用方法请参考其声明.

## 4.1.1 GPIO 驱动

从 1.02 版本起，原 sam4a_gpio.ko 驱动被去除，在 Linux 系统下 GPIO 操作将由 GPIOLib 完成. gpiolib 库是 Linux 的标准 GPIO 接口. 它在/sys/class/gpio 的路径提供一个访问 io 的方法. 注意到在 driver/gpio 的源代码中，gpio-omap.c 为底层接口，gpiolib.c 为实际驱动.

驱动库 sam4a_driver_arm.so 将自动打开(export)其所需要的 GPIO 引脚，用户也可以通过向/sys/class/gpio/gpioXX 中的 direction 和 value 文件写入数据，来控制 GPIO. 具体信息，请参考 GPIO Sysfs Interface for Userspace.

## 4.1.2 SPI 驱动

sam4a_spi.ko 驱动是一个字符设备驱动，Major 为 241,支持设备读写和 ioctl. 关于 SPI 驱动的具体使用，请参考 《SAM4A 编程手册》 第 4.6.2 节.

驱动的文件操作函数声明如下:

```
static int spi_open(struct inode *inode, struct file *file);
static int spi_release(struct inode *inode, struct file *file);
static ssize_t spi_read(struct file *file, char *buf, size_t count, loff_t *offset);
static ssize_t spi_write(struct file *file,const char __user *buf, size_t count, loff_t *offset);
static long spi_ioctl(struct file *f, unsigned int cmd, unsigned long arg);
```

对 SPI Flash 的操作是从一个写命令开始的，写命令将向 SPI 设备发起一个命令，并接收固定长度的数据. 如果写操作返回 1，则为操作成功. 此时用户可以通过读命令读取刚才操作中接收的数据. 如果读取数据长度小于实际读取长度， 数据保持不变， 可以继续读取。如果大于等于实际读取长度， 则数据被清空直到下一次写入。

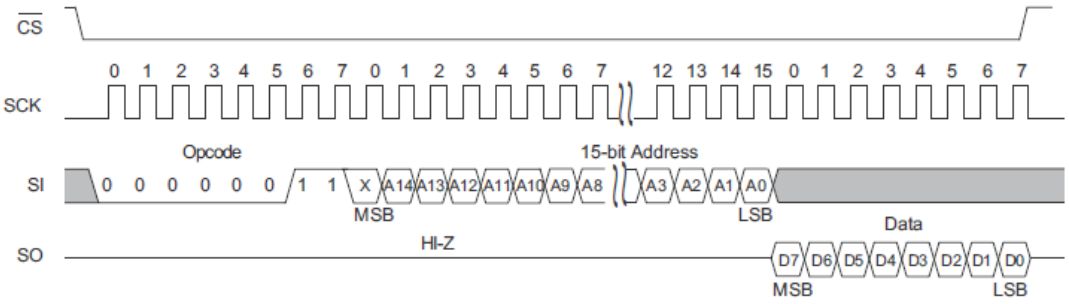
片选的范围由 0 至 4，其具体对应芯片如下：

| 片选 CS | 控制芯片 |
|---|---|
| 0x00 | FERAM 存储器 |
| 0x01 | HF 通道 1，RC531 |
| 0x02 | HF 通道 2，RC531 |
| 0x03 | UHF 通道，Zigbee |
| 0x04 | LED 数码管驱动芯片 |

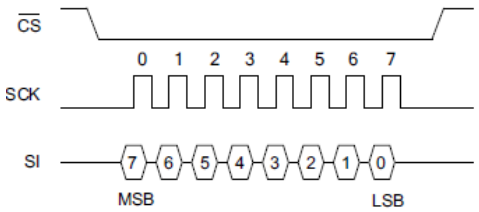
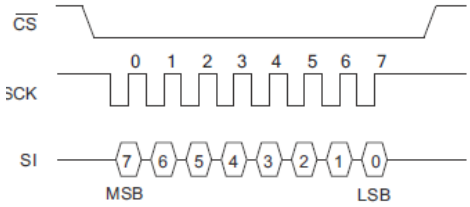
写命令的格式如下表.

| 当前 CS | 写入数据个数 | 读取数据个数 | 写入数据 |
|---|---|---|---|
| 0x00-0x04 | 2 字节 | 2 字节 | 可变长 |

我们以 FERAM 芯片的控制时序说明该驱动的调用方法，




上图为读取 FERAM 数据需要产生的时序，读取地址为 0x0123 中数据所对应的写函数缓冲区数据为 0x00 0003 0001 030123. 即片选为 0，写入 3 字节，读取 1 字节，写入三字节的内容为 0x03(读命令) 0123(读取地址).注意到由于内核内存限制，无法一次性写入大于 512 字节的命令.IO 控制函数命令如下表.用户可以调用 ioctl 命令，设置 SPI 外设的频率和模式.

| Cmd 参数 | Arg 参数 | 描述 |
|---|---|---|
| IOCTL_SPI_MODE | 0 | SPI 模式 0  |
| | 3 | SPI 模式 3  |

| | | |
|---|---|---|
| IOCTL_FREQ | 0x01 | SPI 时钟为 12MHz |
| | 0x02 | SPI 时钟为 6MHz |
| | 0x03 | SPI 时钟为 3MHz |
| | 0x07 | SPI 时钟为 64KHz |

## 4.2 应用程序接口

libsam4a_driver_arm.so 是一个读卡机各个子系统的驱动动态库，用户可以动态连接到该库，并直接调用其函数. 出于兼容性考虑，该动态库提供了两组函数声明,用户可以使用任何一组，函数的实现方法是完全相同的.推荐用户使用以 SAM4A_开头的函数声明. 关于该动态库的链接和编译使用，请参考《SAM4A 编程手册》 第 3 章.

### 4.2.1 通用函数

为了便于用户使用读卡机，通用函数定义了一些数据转换等常用的函数.

| | |
|---|---|
| 函数原型 | uint64_t SAM4A_Utility_GetFileLength(FILE* pFile) |
| 等效函数原型 | long GetFileLength( IN FILE *fp ) |
| 参数 | pFile 为文件指针 |
| 返回值 | 64 位长文件长度，失败返回 0. |
| 描述 | 该函数返回对应文件的长度，文件指针必须指向一个打开的，具有读访问权限的文件.<br>函数调用之后，文件定位器不会变化. 该函数不会关闭文件指针，调用 fclose 需要由用户完成. |

| | |
|---|---|
| 宏定义 | SAM4A_Utility_mSleep(nMilliseconds) |
| 等效宏定义 | Sleep(dwMilliseconds ) |
| 描述 | #define SAM4A_Utility_mSleep(nMilliseconds)<br>usleep(nMilliseconds*1000) |

| | |
|---|---|
| 函数原型 | uint8_t SAM4A_Utility_BinToBCD(uint8_t nBin) |
| 等效函数原型 | unsigned char bin2bcd(unsigned char pBin) |
| 参数 | nBin 为二进制数据 |
| 返回值 | 转换成的 BCD 码. |

| | |
|---|---|
| 函数原型 | uint8_t SAM4A_Utility_BCDToBin(uint8_t nBCD) |
| 等效函数原型 | unsigned char bcd2bin(unsigned char pbcd) |
| 参数 | nBCD 为 BCD 码 |
| 返回值 | 转换成的二进制数据. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Utility_Little_htoi(uint8_t* pHex) |
| 等效函数原型 | unsigned long char2long(unsigned char *msb_mlb) |
| 参数 | pHex 为需要转化的 4 个字节数据 |
| 返回值 | 转换成的整形数据结果，小尾端格式. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Utility_Big_htoi(uint8_t* pHex) |
| 等效函数原型 | unsigned long fchar2long(unsigned char *msb_mlb) |
| 参数 | pHex 为需要转化的 4 个字节数据 |
| 返回值 | 转换成的整形数据结果，大尾端格式. |

| | |
|---|---|
| 函数原型 | void SAM4A_Utility_Little_itoh(uint32_t nInt,uint8_t* pHex) |
| 等效函数原型 | void long2char(unsigned long dwData, unsigned char *msb_mlb) |
| 参数 | nInt 为需要转化的 32 位数据，小尾端格式<br>pHex 为 4 个字节数据地址 |
| 返回值 | 无 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Utility_GetRandom(void) |
| 参数 | 无 |
| 返回值 | 4 字节随机数 |
| 描述 | 该函数通过 AM3552 处理器的硬件随机数产生器生成一个随机数注意到从硬件生成的随机数与 C 库中的伪随机数有着本质的区别，且不需要设置种子 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Utility_GetRandomBuffer(uint8_t* pBuffer,uint32_t nLength) |
| 参数 | pBuffer 为随机数数据地址<br>nLength 为随机数数据长度 |
| 返回值 | 实际生成的随机数长度 |
| 描述 | 该函数通过 AM3552 处理器的硬件随机数产生器生成一个随机数数组. 注意到从硬件生成的随机数与 C 库中的伪随机数有着本质的区别，且不需要设置种子 |

### 4.2.2 CRC 校验函数

CRC 校验函数支持 6 种 CRC16 算法,1 种 CRC32 算法和 DES 加密算法. CRC 算法的宏定义如下表:

| CRC 类型 | 描述 | 宏定义 | 值 |
|---|---|---|---|
| CRC-16-ANSI 多项式为 0x8005 | | | |
| 初始值为 0x0000 | 称为 CRC-IBM 或 CRC-16/ARC | SAM4A_CHECKSUM_CRC16ANSI_IBM | 1 |
| 初始值为 0xFFFF | 用在 Modbus 协议中 | SAM4A_CHECKSUM_CRC16ANSI_MODBUS | 2 |
| CRC-16-CCITT 多项式为 0x1021 | | | |
| 初始值为 0x0000 | 用在 XModem 协议中 | SAM4A_CHECKSUM_CRC16CCITT_XMOD | 3 |
| 初始值为 0xFFFF | 用在 EPCGen2 RFID 卡片中 | SAM4A_CHECKSUM_CRC16CCITT_EPC | 4 |
| 初始值为 0x1D0F | 用在 Fujitsu 芯片的 SPI 协议中 | SAM4A_CHECKSUM_CRC16CCITT_FUJITSU | 5 |
| Kermit 算法 | 又称 CRC-16/CCITT-TRUE | SAM4A_CHECKSUM_CRC16CCITT_KERMIT | 6 |
| CRC-32 多项式为 0x04C11DB7 | | | |
| 初始值为 0xFFFFFFFF | | SAM4A_CHECKSUM_CRC32 | 7 |

DES 加密算法的宏定义如下, 分别为 DES 加密,解密和三重 DES 加密,解密

```
#define SAM4A_CHECKSUM_DES_ENCRYPTIOIN 12
#define SAM4A_CHECKSUM_DES_DECRYPTIOIN 13
#define SAM4A_CHECKSUM_TRIPLE_DES_ENCRYPTIOIN 14
#define SAM4A_CHECKSUM_TRIPLE_DES_DECRYPTIOIN 15
```

| 函数原型 | void SAM4A_CRC_Init(uint8_t nChecksumType,uint32_t* nChecksum, uint8_t* pTableBuffer) |
|---|---|
| 参数 | nChecksumType 为校验值类型, 由宏 SAM4A_CHECKSUM_CRC16ANSI_IBM 等定义.<br>nChecksum 为校验值的数据. 对于 CRC 校验算法,为一个 32 位整型.对于 DES 加密算法, 为一个 32 位整型数组的指针,DES 算法数组长度为 32, 三重 DES 为 96.<br>pTableBuffer 当计算 CRC32 时, pTableBuffer 需要指向一个 1K 字节的缓冲区. 注意: 这个 1K 字节缓冲区的地址必须 4 字节对齐. 但是当 CRC32 的表已经被初始化后, 该参数可以为 NULL, 避免重复初始化. 在计算 CRC16 时, pTableBuffer 可以为 NULL.<br>在计算 DES 和三重 DES 加密时,该参数为秘钥数组,DES 算法秘钥为 8 字节, 三重 DES 加密秘钥为 24 字节. |
| 返回值 | 无 |
| 描述 | 该函数初始化 CRC 校验的映射表和初始值 |

| | |
|---|---|
| 函数原型 | void SAM4A_CRC_AddByte(uint8_t nChecksumType,uint32_t* nChecksum, uint8_t nData) |
| 参数 | nChecksumType 为校验值类型, 由宏 SAM4A_CHECKSUM_CRC16ANSI_IBM 等定义.<br>nChecksum 为校验值的数据.<br>nData 为添加校验的数据 |
| 返回值 | 无 |
| 描述 | 该函数向当前的校验值添加一个字节,该函数不支持 DES 和三重 DES 校验算法. |

| | |
|---|---|
| 函数原型 | void SAM4A_CRC_AddBuffer(uint8_t nChecksumType,uint32_t* nChecksum,uint8_t* pData,uint32_t nLength) |
| 参数 | nChecksumType 为校验值类型, 由宏 SAM4A_CHECKSUM_CRC16ANSI_IBM 等定义.<br>nChecksum 为校验值的数据. 对于 CRC 校验算法,为一个 32 位整型.对于 DES 加密算法, 为一个 32 位整型数组的指针,DES 算法数组长度为 32, 三重 DES 为 96.<br>pData 为校验数据缓冲区<br>nLength 为校验数据缓冲区长度,如果为 DES 加密算法, 数据的长度必须为 8 字节对齐. |
| 返回值 | 无 |
| 描述 | 该函数向当前的校验值添加一个缓冲区 |

| | |
|---|---|
| 函数原型 | void SAM4A_CRC_GetKermitResult(uint32_t* nChecksum) |
| 参数 | nChecksum 为校验值的数据. |
| 返回值 | 无 |
| 描述 | 该函数计算 Kermit 的校验结果 |

| | |
|---|---|
| 函数原型 | void SAM4A_CRC_GetCRC32Result(uint32_t* nChecksum) |
| 参数 | nChecksum 为校验值的数据. |
| 返回值 | 无 |
| 描述 | 该函数计算 CRC32 的校验结果 |

### 4.2.3 加密函数

读卡机的加密函数提供以下几种算法: DES, TripleDES, RSA, MD5, SHA 1, SHA224, SHA256, SHA384, SHA512.其中,DES 加密算法的使用,请参考 4.2.2 节. RSA 和 SHA 加密算法使用了 OpenSSL 的加密库, 如果需要调用这些函数, 则还需要连接到 libcrypto.so.

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RSA_PublicDecryption(uint8_t* pPublicKey,uint32_t nPublicKeyLength,uint32_t e,uint8_t* pFrom,uint8_t* pTo,uint32_t nLength) |
| 参数 | pPublicKey 为公共密钥缓冲区指针<br>nPublicKeyLength 为公共密钥长度<br>e 为生成公钥的指数<br>pFrom 为输入数据指针<br>pTo 为输出数据指针<br>nLength 为输入数据长度 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数首先生成一个公钥, 使用这个公钥进行解密 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RSA_PublicEncryption(uint8_t* pPublicKey,uint32_t nPublicKeyLength,uint32_t e,uint8_t* pFrom,uint8_t** pTo,uint32_t nLength) |
| 参数 | pPublicKey 为公共密钥缓冲区指针<br>nPublicKeyLength 为公共密钥长度<br>e 为生成公钥的指数<br>pFrom 为输入数据指针<br>pTo 为输出数据指针<br>nLength 为输入数据长度 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数首先生成一个公钥, 使用这个公钥进行加密 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RSA_PrivateDecryption(uint8_t* pPublicKey,uint32_t nPublicKeyLength,uint32_t e,uint8_t* pPrivateKey,uint32_t nPrivateKeyLength, uint8_t* pFrom,uint8_t* pTo,uint32_t nLength) |
| 参数 | pPublicKey 为公共密钥缓冲区指针<br>nPublicKeyLength 为公共密钥长度<br>e 为生成公钥的指数<br>pPrivateKey 为私钥缓冲区指针<br>nPrivateKeyLength 为私钥长度 |

| | |
|---|---|
| | pFrom 为输入数据指针<br>pTo 为输出数据指针<br>nLength 为输入数据长度 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | RSA 算法私钥解密函数 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RSA_PrivateEncryption(uint8_t*<br>pPublicKey,uint32_t nPublicKeyLength,uint32_t<br>e,uint8_t* pPrivateKey,uint32_t nPrivateKeyLength,<br>uint8_t* pFrom,uint8_t** pTo,uint32_t nLength) |
| 参数 | pPublicKey 为公共密钥缓冲区指针<br>nPublicKeyLength 为公共密钥长度<br>e 为生成公钥的指数<br>pPrivateKey 为私钥缓冲区指针<br>nPrivateKeyLength 为私钥长度<br>pFrom 为输入数据指针<br>pTo 为输出数据指针<br>nLength 为输入数据长度 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | RSA 算法私钥加密函数 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Digestion(uint8_t nDigestAlgorithm,const<br>uint8_t* pSource,uint32_t nSourceLength,uint8_t*<br>pDigestion) |
| 等效函数原型 | int SHA_1(const unsigned char *pIndata, int InLen,<br>unsigned char *md)<br>int SHA_224(const unsigned char *pIndata, int InLen,<br>unsigned char *md)<br>int SHA_256(const unsigned char *pIndata, int InLen,<br>unsigned char *md)<br>int SHA_384(const unsigned char *pIndata, int InLen,<br>unsigned char *md)<br>int SHA_512(const unsigned char *pIndata, int InLen,<br>unsigned char *md) |
| 参数 | nDigestAlgorithm 为摘要算法定义,可以为下列值之一<br>#define SAM4A_CHECKSUM_MD5 1<br>#define SAM4A_CHECKSUM_SHA1 2<br>#define SAM4A_CHECKSUM_SHA224 3 |

| | |
|---|---|
| | #define SAM4A_CHECKSUM_SHA256 4<br>#define SAM4A_CHECKSUM_SHA384 5<br>#define SAM4A_CHECKSUM_SHA512 6<br>pSource 为输入数据指针<br>nSourceLength 为输入数据长度<br>pDigestion 为输出摘要值 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 摘要算法函数 |

### 4.2.4 蜂鸣器函数

控制蜂鸣器的引脚默认处在输入状态，只有打开蜂鸣器函数执行后，才会将引脚置为输出，并控制蜂鸣器鸣响. 注意到 Beep 函数在调用时处在阻塞状态，即在蜂鸣器鸣响完后，函数才会返回.

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Buzzer_Open(void) |
| 等效函数原型 | uint32_t BeepKeyOpen(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数打开蜂鸣器，将引脚置为输出状态 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Buzzer_Close(void) |
| 等效函数原型 | uint32_t BeepKeyClose(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数关闭蜂鸣器，将引脚置为输入状态 |

| | |
|---|---|
| 函数原型 | void SAM4A_Buzzer_Beep(uint32_t nDelay) |
| 等效函数原型 | void Beep(uint32_t Delays) |
| 参数 | 蜂鸣器鸣响时间，以 ms 为单位. |
| 返回值 | 无 |
| 描述 | 该函数在调用时处在阻塞状态，即在蜂鸣器鸣响完后，函数才会返回 |

### 4.2.5 按键函数

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Button_GetStatus(void) |
| 等效函数原型 | int GetKey(void) |
| 参数 | 无 |
| 返回值 | 1 为按键已按下，0 为未按下 |
| 描述 | 按键引脚永远处于输入状态，调用该函数前无需初始化 LED 模块．该函数有防干扰功能，在 0.1 秒内会多次检查引脚状态，每次处在按下状态时才会返回 1. |

### 4.2.6 以太网函数

尽管驱动提供了一组以太网控制函数，建议用户在编程时使用 Linux 的标准套接字函数，使得代码具有可移植性.本质上,下面的以太网函数只是在调用 Linux 定义的 socket 函数.

| | |
|---|---|
| 函数原型 | int32_t SAM4A_Socket_Open(char* pDevice,char* pIPAddress,uint32_t nMode,uint32_t nTimeout,uint32_t* pErrorCode) |
| 等效函数原型 | int TCPIP_CommOpen(char *pPortDescriptor,char *pOpenParams,unsigned int dwPortAttr,int nTimeout,int *pErrCode ) |
| 参数 | pDevice 为网络设备描述符<br>pIPAddress 为 IP 地址和端口号字符串<br>nMode 为工作模式<br>nTimeout 为超时时间<br>pErrorCode 为返回错误代码 |
| 返回值 | 返回套接字句柄，如果为-1 则失败，其他则成功. |
| 描述 | 该函数创建套接字,设置套接字的 IP 地址和端口并连接到服务器(如果是客户端)．即该函数完成 socket 和 connect 两个函数的功能. |

| | |
|---|---|
| 函数原型 | int32_t SAM4A_Socket_Accept(int32_t nSocketHandle) |
| 等效函数原型 | int TCPIP_CommAccept( int hPort ) |
| 参数 | nSocketHandle 为当前套接字句柄 |
| 返回值 | 返回套接字句柄，如果为-1 则失败，其他则成功. |
| 描述 | 该函数被服务器调用，用于监听客户端．返回对应客户端的套接字句柄．该函数等效于调用 accept(). |

| | |
|---|---|
| 函数原型 | int32_t SAM4A_Socket_Read(int32_t nSocketHandle,uint8_t* pBuffer, int32_t nBytesToRead) |
| 等效函数原型 | int TCPIP_CommRead(int hPort,char *pBuffer,int nBytesToRead) |
| 参数 | nSocketHandle 为当前套接字句柄<br>pBuffer 为接收数据指针<br>nBytesToRead 为接受数据长度 |
| 返回值 | 返回实际从套接字读取的数据长度. |
| 描述 | 该函数等效于调用 recv (). |

| | |
|---|---|
| 函数原型 | int32_t SAM4A_Socket_Write(int32_t nSocketHandle, uint8_t* pBuffer, int32_t nBytesToWrite) |
| 等效函数原型 | int TCPIP_CommWrite(int hPort,char *pBuffer,int nBytesToWrite) |
| 参数 | nSocketHandle 为当前套接字句柄<br>pBuffer 为发送数据指针<br>nBytesToWrite 为发送数据长度 |
| 返回值 | 返回实际从套接字发送的数据长度. |
| 描述 | 该函数等效于调用 send(). |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Socket_Control(int32_t nSocketHandle, int32_t nCmd, uint8_t* pBuffer, int32_t nDataLength) |
| 等效函数原型 | int TCPIP_CommControl(int hPort, int nCmd,void *pBuffer,int nDataLength) |
| 参数 | nSocketHandle 为当前套接字句柄<br>nCmd 为命令字节<br>pBuffer 为数据指针<br>nDataLength 为数据长度 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 通过套接字发送控制命令 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Socket_Close(int32_t nSocketHandle) |
| 等效函数原型 | int TCPIP_CommClose(int hPort) |
| 参数 | nSocketHandle 为当前套接字句柄 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 关闭套接字, 该函数等效于调用 shutdown ()和 close(). |

### 4.2.7 串口函数

和以太网控制函数类似，我们建议用户在编程时使用 Linux 的标准串口函数，使得代码具有可移植性．注意到在当前 Linux 环境下，串口分配如下表所示，串口的位置参考 2.5 节．如有需要用户可以通过 mknod 命令将对应串口映射到其他地址．如命令 mknode /dev/ttyS4 c 250 1 将串口 0 映射到 ttyS4 上．

| 插座 | 信号 | 芯片外设 | Linux 设备 | Major | Minor |
|---|---|---|---|---|---|
| 串口 0(J9) | TXD1/RXD1 | UART1 | /dev/tty01 | 250 | 1 |
| 串口 1(J10) | TXD2/RXD2 | UART4 | /dev/tty04 | 250 | 4 |
| 串口 2(J11) | TXD3/RXD3 | UART0 | /dev/tty00 | 250 | 0 |

| | |
|---|---|
| 函数原型 | int32_t SAM4A_Serial_Open(char* pDevice,char* pParameter,uint32_t nMode,uint32_t nTimeout,uint32_t* pErrorCode) |
| 等效函数原型 | int Serial_CommOpen(char *pPortDescriptor,char *pOpenParams,unsigned int dwPortAttr,int nTimeout,int *pErrCode ) |
| 参数 | pDevice 为串口设备描述符<br>pParameter 为串口的波特率和其他参数<br>nMode 为工作模式<br>nTimeout 为超时时间<br>pErrorCode 为返回错误代码 |
| 返回值 | 返回串口句柄，如果为-1 则失败，其他则成功. |
| 描述 | 该函数创建串口句柄,设置其参数并打开串口．即该函数完成 open 和 tcsetattr 两个函数的功能. |

| | |
|---|---|
| 函数原型 | int32_t SAM4A_Serial_Read(int32_t nHandle, uint8_t* pBuffer, int32_t nBytesToRead) |
| 等效函数原型 | int Serial_CommRead(int hPort,char *pBuffer,int nBytesToRead) |
| 参数 | nHandle 为当前串口句柄<br>pBuffer 为接收数据指针<br>nBytesToRead 为接受数据长度 |
| 返回值 | 返回实际从串口读取的数据长度. |
| 描述 | 该函数等效于调用 read ( ). |

| | |
|---|---|
| 函数原型 | int32_t SAM4A_Serial_Write(int32_t nHandle, uint8_t* pBuffer, int32_t nBytesToWrite) |
| 等效函数原型 | int Serial_CommWrite(int hPort,char *pBuffer,int nBytesToWrite) |
| 参数 | nHandle 为当前串口句柄<br>pBuffer 为发送数据指针<br>nBytesToWrite 为发送数据长度 |
| 返回值 | 返回实际从串口发送的数据长度. |
| 描述 | 该函数等效于调用 write(). |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Serial_Control(int32_t nHandle, int32_t nCmd, uint8_t* pBuffer, int32_t nDataLength) |
| 等效函数原型 | int Serial_CommControl(int hPort, int nCmd,void *pBuffer,int nDataLength) |
| 参数 | nHandle 为当前串口句柄<br>nCmd 为命令字节<br>pBuffer 为数据指针<br>nDataLength 为数据长度 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 通过串口发送控制命令 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Serial_Close(int32_t nHandle) |
| 等效函数原型 | int Serial_CommClose(int hPort) |
| 参数 | nHandle 为当前串口句柄 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 关闭串口, 该函数等效于调用 close(). |

### 4.2.8 HF 射频函数

用户首先需要初始化 HF 射频基带芯片，然后执行对应的操作. 其基本流程由由 ISO14443-4 标准定义，详细的使用方法，请参考 《SAM4A 编程手册》 第 3.7 节.

以 SAM4A_开头的函数可以选择当前操作的通道，推荐用户使用. 而以 uRf_和 sRf_为前缀的宏定义则用于提供一个兼容的接口.

例如，我们定义了 #define uRf_Init() (int)SAM4A_RFID_Init(0)

于是调用 uRf_Init()就等效于调用 SAM4A_RFID_Init(0).

HF 函数列表如下，注意到有些函数需要针对特定的卡片调用.

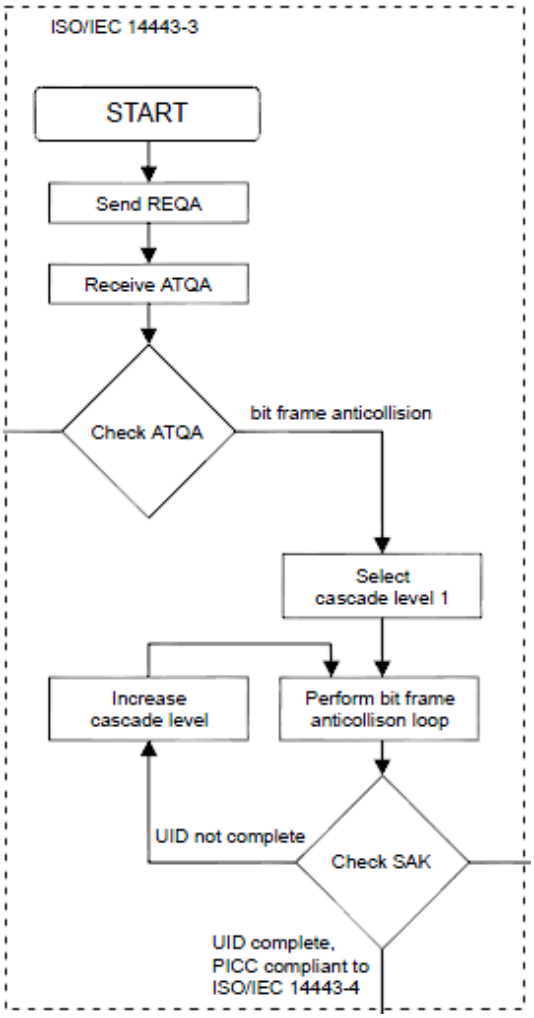
| 函数名 | 等效函数名 | 描述 | Mifare 卡 | CPU 卡 | Ultralight 卡 |
|---|---|---|---|---|---|
| TypeA/B | | | | | |
| SAM4A_RFID_Init | uRf_Init | 初始化 | √ | √ | √ |
| SAM4A_RFID_TxOnOff | uRf_OnOff | 关闭射频输出 | √ | √ | √ |
| SAM4A_RFID_AutoSleep | 无 | 自动休眠 | √ | √ | √ |
| SAM4A_RFID_SetEncryptionKeySet | 无 | 设置秘钥集 | √ | | |
| SAM4A_RFID_ReadEEROM | 无 | 读取 EEROM | √ | √ | √ |
| SAM4A_RFID_WriteEEROM | 无 | 写入 EEROM | √ | √ | √ |
| SAM4A_RFID_Halt | uRful_Halt | 中止卡片 | √ | √ | √ |
| TypeA | | | | | |
| SAM4A_RFID_TypeA_Request | uRfa_GetSNR | 寻卡 | √ | √ | √ |
| SAM4A_RFID_RATS | uRfa_GetCardDetails | CPU 卡初始化 | | √ | |
| SAM4A_RFID_APDUTranceive | uRfa_APDU | 发送 APDU 命令 | | √ | |
| SAM4A_RFID_Authentication | uRfulc_Authen | 三重认证 | √ | | √ |
| SAM4A_RFID_Read_Protection | 无 | 带保护读卡 | √ | | √ |
| SAM4A_RFID_Write_Protection | 无 | 带保护写卡 | √ | | √ |
| SAM4A_RFID_Read_NoProtection | sRf_Read | 读卡 | √ | | √ |
| SAM4A_RFID_Write_NoProtection | uRful_Write | 写卡 | √ | | √ |
| SAM4A_RFID_InitPurse | 无 | 初始化值块 | √ | | |
| SAM4A_RFID_ReadPurse | 无 | 读取值块 | √ | | |
| SAM4A_RFID_PurseIncrement | uRfmif_IncTransfer | 值块加 | √ | | |
| SAM4A_RFID_PurseDecrement | uRfmif_DecTransfer | 值块减 | √ | | |
| SAM4A_RFID_PurseBackup | uRfmif_RestoreTran | 备份值块 | √ | | |
| TypeB | | | | | |
| SAM4A_RFID_TypeB_Request | 无 | 寻卡 | | | |
| SAM4A_RFID_TypeB_Attribute | 无 | 归属 | | | |
| SAM4A_RFID_TypeB_Read_Protection | 无 | 读卡 | | | |

| | | | | | |
|---|---|---|---|---|---|
| SAM4A_RFID_TypeB_Write_Protection | 无 | 写卡 | | | |
| SAM4A_RFID_TypeB_SetUserZone | 无 | 设置用户区 | | | |
| SAM4A_RFID_TypeB_VerifyCrypto | 无 | 验证加密命令 | | | |
| SAM4A_RFID_TypeB_SendChecksum | 无 | 发送校验和 | | | |
| SAM4A_RFID_TypeB_Idle | 无 | 空闲命令 | | | |
| SAM4A_RFID_TypeB_CheckPassword | 无 | 检测密钥命令 | | | |

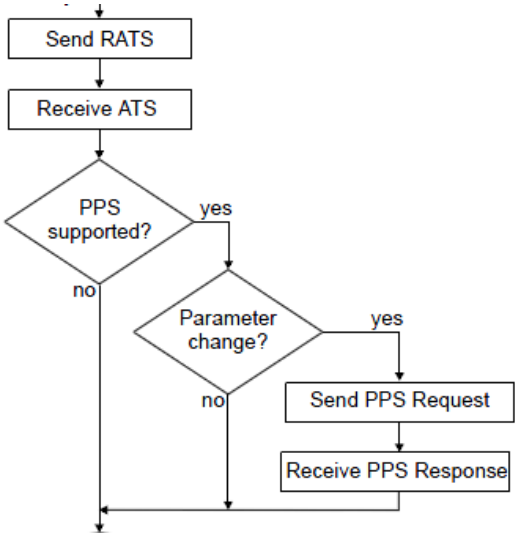
| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_Init(uint8_t nChannel) |
| 等效函数原型 | int uRf_Init(void)<br>int RFIDInit (void) |
| 参数 | nChannel 为射频通道号，0-1 有效 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 初始化 RFID 射频基带芯片的寄存器，此时射频输出为关闭状态. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TxOnOff(uint8_t nChannel,uint32_t nTxOn) |
| 等效函数原型 | int uRf_OnOff(unsigned int mode)<br>int sRf_OnOff(unsigned int mode) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nTxOn 为 1 则打开射频输出，0 则关闭射频 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 控制射频输出状态.<br>**注意：**考虑到降低功放功率,该函数打开射频输出功能被禁止,即只能关闭射频. 而打开射频的操作由寻卡函数自动完成. 注意到除了 TxOnOff，休眠函数 Halt 也会关闭射频输出. 当操作超时后,系统(SAM4A_RFID_AutoSleep)也会自动关闭射频. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeA_Request(uint8_t nChannel, uint8_t nRequestAll,uint8_t* pLength, uint8_t* pRequestData) |
| 等效函数原型 | int uRfa_GetSNR (unsigned char mode, unsigned char* len, unsigned char* UID)<br>int sRf_GetSNR (unsigned int mode, unsigned char* bLen, unsigned char* pSNR) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nRequestAll 为寻卡模式,0 是 RequestStd 执行 REQA(0x26)，1 是 RequestAll 执行 WUPA(0x52)命令<br>pLength 为返回数据总长度<br>pRequestData 为返回数据，即 UID(4/7/10 字节) +SAK(1 个字节)+ ATQA(2 字节) |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数会查询某射频通道是否有卡存在，如果有卡，则该命令返回成功，用户通过卡片返回的 SAK 和 UID 等信息来判断下面的操作.该命令执行由 ISO14443-3 定义的寻卡和防重叠环的操作. |

| | |
|---|---|
| | 其流程图如下.<br>  <br><br>在返回的数据中,ATQA 为卡片响应 REQA 或 WUPA 命令返回的 2 字节数据，SAK 为卡片响应选卡命令返回的 1 字节数据，UID 为卡片内部 ID 号．这三个参数的定义请参考 ISO14443 标准. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_RATS(uint8_t nChannel,uint8_t nExpectedDataRate,uint8_t* pCurrentDataRate,uint16_t* pATQA,uint8_t* pSAK,uint8_t* pATS) |
| 等效函数原型 | int uRfa_GetCardDetails(unsigned short* ATQA,unsigned char* SAK,unsigned char* ATS)<br>int sRfa_RATS(OUT INT8U *len,OUT INT8U *OutData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nExpectedDataRate 为期望 CPU 卡通信速率,其末尾 2 位(LSB,LSB+1)为 DRI，即从 RC531 到卡片的 bit 传输速率．第 2-3 位(LSB+2,LSB+3)为 DSI，即从卡片到 RC531 的 bit 传输速率．如 |

| | |
|---|---|
| | 果值为 00，速率为 1；01 速率为 2；10 速率为 4；11 速率为 8. 分别对应着 106Kbps，212Kbps，424Kbps 和 848Kbps 传输速率. 最高位如果为 1,则不执行 PPS. 例如，0x05 表示期望上行和下行传输速率均为 212KBps. 0x80 表示不进行 PPS.<br>pCurrentDataRate 为实际 CPU 卡通信速率,格式同上. 如果该指针为 NULL，则不返回数据.<br>pATQA 指向寻卡过程中返回的 ATQA 值<br>pSAK 指向寻卡过程中返回的 SAK 值<br>pATS 指向卡片响应 RATS 返回的数据，其实际长度由第一个字节定义，具体 ATS 数据结构,请参考 ISO14443-4 标准，5.2 节. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 在完成寻卡后，如果卡片属于 CPU 卡，则应该调用该函数进行 CPU 卡的初始化. 该函数主要执行由 ISO14443-4 标准定义的 RATS 和 PPS 两个命令. 在 CPU 卡初始化之后，用户就可以通过 APDU 命令与卡进行通信.函数流程图如下.<br>  <br><br>**注意：** 参数 nExpectedDataRate 仅仅是期望使用的速率，CPU 卡在 ATS 的 TA(1)字节指示了当前卡支持的速率，读卡机会根据卡支持速率自动选择最接近的值.例如如果用户期望上行和下行传输速率均为 212KBps，但卡片只支持 106KBps，则读卡机会选择 106KBps 作为 PPS 命令参数.参数 pCurrentDataRate 返回的值为 0x00. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_APDUTranceive(uint8_t nChannel,uint8_t* pTransmitBuffer,uint32_t nTransmitCount,uint8_t* pReceiveBuffer,uint32_t* pReceiveCount,uint8_t nAPDUFlag,uint16_t* pSW) |
| 等效函数原型 | int uRfa_APDU(unsigned char*send,unsigned int len,unsigned char* OutData,unsigned int*OutLen,unsigned short*SW)<br>int uRfdes_APDU(unsigned char*send,unsigned int len,unsigned char*OutData,unsigned char*OutLen)<br>int sRfa_APDU(unsigned char*send,unsigned int len, unsigned char*OutData,unsigned int*OutLen,unsigned short*SW)<br>int sRfdes_APDU(unsigned char*send,unsigned int len,unsigned char*OutData,unsigned int*OutLen) |
| 参数 | nChannel 为射频通道号, 0-1 有效<br>pTransmitBuffer 为发送 APDU 数据包缓冲区指针, CPU 卡的 APDU 命令格式由 ISO7816 标准定义,调制芯片的命令缓冲区长度为 512 字节, 这意味着用户最大可以发送的 APDU 命令长度为 490 字节.<br>nTransmitCount 为发送 APDU 数据包长度<br>pReceiveBuffer 为接收 APDU 数据包缓冲区指针,<br>与参数 pTransmitBuffer 可以指向同一个缓冲区<br>pReceiveCount 为接收 APDU 数据包长度指针, 如果返回成功,该长度有效.<br>nAPDUFlag 为 APDU 数据包发送标志位, 如果最低位为 1, 则发送的数据包头中含有 CID 码.<br>pSW 为接收 APDU 数据包的前两个字节, 如果不为空指针,则 pReceiveBuffer 指向返回数据的第三个字节. 如果为空指针,则 pReceiveBuffer 指向返回数据的第一个字节. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 在完成 CPU 卡初始化后, 用户可以执行该函数向 CPU 卡发送 APDU 命令并接受 APDU 命令响应.传输协议使用 ISO14443-4 中定义的半双工块传输协议(T=1). |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_Halt(uint8_t nChannel,uint8_t nTagType) |
| 等效函数原型 | void uRful_Halt(void)<br>int uRfa_Halt(void)<br>void sRfa_Halt (void) |

| | |
|---|---|
| 参数 | nChannel 为射频通道号，0-1 有效<br>nTagType 为卡片类型,内容如下<br>0x00　　为符合 ISO14443 TypeA，Mifare 卡<br>0x01　　为符合 ISO14443 TypeA，CPU 卡<br>0x02　　为符合 ISO14443 TypeB 卡 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该命令将卡片置入 Halt 状态，如果当前卡为 Mifare 卡，则使用 HALT 命令．如果当前卡为 CPU 卡或 TypeB 卡，则使用对应的 DESELECT 命令．该命令还会关断当前通道的射频信号．<br>**注意：**在卡片完成操作之后，正常情况下用户应该调用 Halt 函数以停止卡片操作并关闭射频．如果用户在完成操作之后不调用该函数，或是 TxOnOff，超出一段时间后,驱动库将自行关闭射频输出．请参考函数 SAM4A_RFID_AutoSleep |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_Authentication(uint8_t nChannel,uint16_t nKey,uint8_t nSectorAddress,unsigned char* pKeyValue,uint8_t nAuthenticationFlag) |
| 等效函数原型 | int uRfulc_Authen(unsigned short wKeyNumber,unsigned short wKeyVersion)<br>int uRfmif_Authen(unsigned char cKeyab,unsigned char cSecotrNo,unsigned char*pKey,unsigned char*pSNR)<br>int sRf_Authen(unsigned char cKeyab,unsigned char cSecotrNo,unsigned char *pKey,unsigned char*pSNR) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nKey 为 2 字节密钥信息，其最高位如果为 0,表示这个扇区的密钥在 pKeyValue 指向的缓冲区中.0x0001/0x0060/0x000A 表示使用密钥 A，0x0000/0x0061/0x000B 表示使用密钥 B.最高位如果为 1，则后 9 位表示密钥在 EEROM 中的地址.此时,次高位(MSB-1)为 1 表示密钥 A,为 0 表示密钥 B.<br>例如，0xC110 表示密钥为以 0x0110 地址的起始的调制芯片的 EEROM 中，一共 6 个字节,密钥类型为密钥 A．0x0000 表示密钥在 pKeyValue 中,为密钥 B.<br>nSecotrAddress 为三重认证的扇区地址，范围从 0x00 至 0x37，支持最大 16Kbytes 的 Mifare 卡.<br>pKeyValue 为秘钥值，当 nKey 的最高位为 1 时，可以为 NULL<br>nAuthenticationFlag 为三重认证标志位<br>0 表示进行默认 Mifare 卡三重认证<br>1 表示进行 Ultralight 卡认证，例如 FM11RF005 型卡片，从卡片 |

| | |
|---|---|
| | 的 Block0 中读取 4 字节作为卡片 ID,再进行三重认证.Key 对应的 6 字节密钥必须为卡片第 8 个 Block 的 4 字节加上两个 0x00.<br>2 表示跳过三重认证，例如 MF0ICU1 型卡片，UID 的长度为 7 字节，选卡过程和 ISO14443 标准兼容．在完成选卡后并不需要三重认证就能够直接读写.<br>3 表示进行 Ultralight EV1 卡认证， 例如 MF0UL11 型卡片，认证命令为 0x1B．注意：此时 nKey 参数的最高位必须为 0，pKeyValue 指向 4 字节密钥 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数进行 Mifare 卡和 Ultralight 卡的三重认证算法，当 nKey 的最高位为 0 时，pKeyValue 包含 6 字节的密钥数据．然而，为了保证密钥数据的安全性，通常将密钥写入调制芯片的 EEROM 中(参考函数 SAM4A_RFID_WriteEEROM)，而调制芯片的 EEROM 有一部分是 WO 只写属性的，达到保护密钥的目的．此时 nKey 的后 9 位表示密钥在 EEROM 中的地址．但是对于 Ultralight 卡,nAuthenticationFlag 参数为 3 时的操作,则必须将 4 字节密钥放在 pKeyValue 中. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_Read_NoProtection(uint8_t nChannel,uint8_t nReadMode,uint8_t nBlockAddress,uint8_t* pReadLength,uint8_t* pReadData) |
| 等效函数原型 | int sRf_Read (unsigned char cBlockNo,unsigned char*pRdData)<br>int uRful_Read(unsigned char bBock,unsigned char*pData)<br>int uRfmif_Read(unsigned char cBlockNo, unsigned char*pRdData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nReadMode 为读卡模式<br>0 表示 Mifare 卡读取,此时 pReadLength 一定返回 16 字节<br>1 表示 Ultralight 卡读取,此时 pReadLength 可能为 4 或 16 字节<br>nBlockAddress 在 Mifare 模式下为块地址,范围 0x00 至 0xFF．在 Ultralight 模式下为页地址，范围从 0x00 至 0x28，支持最大 40 个页.<br>pReadLength 为读取数据的长度，为空指针则读出默认长度<br>pReadData 为读取数据缓冲区指针，指针指向的缓冲区尺寸必须大于等于 16 字节. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数用于读取 Mifare 卡和 Ultralight 卡，读取的范围包含了 |

| | |
|---|---|
| | Mifare 卡的密钥区，即可以读取出当前 Mifare 卡的密钥. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_Write_NoProtection(uint8_t nChannel,uint8_t nWriteMode,uint8_t nBlockAddress,uint8_t* pWriteData) |
| 等效函数原型 | int uRful_Write(unsigned char bBock,unsigned char*pData)<br>int uRfmif_Write(unsigned char bAddress,unsigned char*pData)<br>int sRf_Write (unsigned char cBlockNo,unsigned char* pWrData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nWriteMode 为写卡模式<br>0 表示写入 Mifare 卡,此时会写入 pWriteData 指向的 16 字节<br>1 表示写入 Ultralight 卡,写卡命令使用 0xA2<br>2 表示写入 Ultralight 卡,写卡命令使用 0xA0，两种 Ultralight 卡写入时均只写入 4 字节.<br>nBlockAddress 在 Mifare 模式下为块地址,范围 0x00 至 0xFF.在 Ultralight 模式下为页地址，范围从 0x00 至 0x28，支持最大 40 个页.<br>pWriteData 为写入数据缓冲区指针 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数用于写入 Mifare 卡和 Ultralight 卡，写入的范围包含了 Mifare 卡的密钥区，即可以设置出当前 Mifare 卡的密钥. |

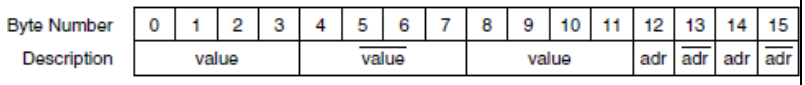
| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_Read_Protection(uint8_t nChannel,uint8_t nReadMode,uint8_t nBlockAddress,uint8_t* pReadLength,uint8_t* pReadData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nReadMode 为读卡模式<br>0 表示 Mifare 卡读取,此时 pReadLength 一定返回 16 字节<br>1 表示 Ultralight 卡读取,此时 pReadLength 可能为 4 或 16 字节<br>nBlockAddress 在 Mifare 模式下为块地址,范围 0x00 至 0xFF.在 Ultralight 模式下为页地址，范围从 0x00 至 0x28，支持最大 40 个页.<br>pReadLength 为读取数据的长度<br>pReadData 为读取数据缓冲区指针，指针指向的缓冲区尺寸必须大于等于 16 字节. |
| 返回值 | 错误码(请参考 4.3 节) |

| | |
|---|---|
| 描述 | 该函数用于读取 Mifare 卡和 Ultralight 卡，读取的数据块地址不能为 Mifare 卡的密钥块. 即当扇区地址在 0x00 至 0x1F 范围内时，读取块地址不能为 0x03. 当扇区地址在 0x20 至 0x27 时，读取块地址不能为 0x0F. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_Write_Protection(uint8_t nChannel,uint8_t nWriteMode,uint8_t nBlockAddress,uint8_t* pWriteData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nWriteMode 为写卡模式<br>0 表示写入 Mifare 卡,此时会写入 pWriteData 指向的 16 字节<br>1 表示写入 Ultralight 卡,写卡命令使用 0xA2<br>2 表示写入 Ultralight 卡,写卡命令使用 0xA0，两种 Ultralight 卡写入时均只写入 4 字节.<br>nBlockAddress 在 Mifare 模式下为块地址,范围 0x00 至 0xFF. 在 Ultralight 模式下为页地址，范围从 0x00 至 0x28，支持最大 40 个页.<br>pWriteData 为写入数据缓冲区指针 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数用于写入 Mifare 卡和 Ultralight 卡，写入的数据块地址不能为卡片的第一个数据块和密钥块. 即当扇区地址为 0x00 时，写入的块地址不能为 0x00 或 0x03. 当扇区地址在 0x01 至 0x1F 范围内时，写入块地址不能为 0x03. 当扇区地址在 0x20 至 0x27 时，写入块地址不能为 0x0F. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_ReadEEROM(uint8_t nChannel,uint16_t nAddress,uint8_t nDataLength,uint8_t* pData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nAddress 为 2 字节写入地址,对于调制芯片 RC531 来说,允许读取的 EEROM 的地址范围从 0x0000 到 0x007F.<br>nDataLength 为读取数据长度，数据长度的范围在 1 到 16 字节.<br>pData 为读取数据指针 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数将普通数据从调制芯片的 EEROM 中读取出来，注意到该函数并不能读取属于秘钥区的数据. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_WriteEEROM(uint8_t nChannel,uint16_t nAddress,uint8_t nKeyFlag,uint8_t nDataLength,uint8_t* pData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nAddress 为 2 字节写入地址,对于调制芯片 RC531 来说，可以写入 EEROM 的地址范围从 0x0000 到 0x01FF.<br>nKeyFlag 为是否以密钥格式写入，如果密钥字节为 0,则直接写入数据. 如果密钥字节为 1,则写入的数据长度必须为 6 字节. 读卡机会将这 6 字节密钥转换成 Crypto1 格式的密钥并写入. 例如，实际密钥 0xA0 A1 A2 A3 A4 A5 将会转换成 0x5A F0 5A E1 5A D2 5A C3 5A B4 5A A5. 关于调制芯片 EEROM 映射和密钥格式，请参考 MFRC500 数据手册 6.1 节和 6.4 节.<br>nDataLength 为写入数据长度，数据长度的范围在 1 到 16 字节.<br>pData 为写入数据指针 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数将普通数据或是密钥写入调制芯片的 EEROM 中，以便在后面需要密钥的函数中直接使用. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_InitPurse(uint8_t nChannel,uint8_t nBlockAddress,uint8_t nAddr,int32_t nValue) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nBlockAddress 为块地址,范围 0x00 至 0xFF.<br>nAddr 为写入值块的地址值<br>nValue 为写入值块的数据值 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 将 Mifare 卡的一个块初始化为一个值块(Value Block)，其具体结构如下图所示.<br>  <br><br>写入的值块地址不能为卡片的第一个数据块和密钥块. 即当扇区地址为 0x00 时,写入的块地址不能为 0x00 或 0x03. 当扇区地址在 0x01 至 0x1F 范围内时，写入块地址不能为 0x03. 当扇区地址在 0x20 至 0x27 时，写入块地址不能为 0x0F. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_ReadPurse(uint8_t nChannel,uint8_t nBlockAddress,uint8_t* pAddr,int32_t* pValue) |
| 参数 | nChannel 为射频通道号，0-1 有效 |

| | |
|---|---|
| | nBlockAddress 为块地址,范围 0x00 至 0xFF.<br>pAddr 指向读取值块的地址值<br>pValue 指向读取值块的数据值 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数从 Mifare 系列卡中读取一个值块(Value Block). 读取的值块地址不能为卡片的密钥块. 即当扇区地址在 0x00 至 0x1F 范围内时, 读取块地址不能为 0x03. 当扇区地址在 0x20 至 0x27 时, 读取块地址不能为 0x0F. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_PurseIncrement(uint8_t nChannel,uint8_t nSourceBlockAddress,uint8_t nDestinationBlockAddress,int32_t nValue) |
| 等效函数原型 | int uRfmif_IncTransfer(unsigned char bSrcBlock,unsigned char bDstBlock,unsigned char*bValue)<br>int sRfmif_IncTransfer(unsigned char bSrcBlock,unsigned char bDstBlock,unsigned char*bValue) |
| 参数 | nChannel 为射频通道号, 0-1 有效<br>nSourceBlockAddress 为源块地址,范围 0x00 至 0xFF.<br>nDestinationBlockAddress 为目标块地址,范围 0x00 至 0xFF.<br>nValue 为向值块中增加的值 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数向 Mifare 系列卡的某一个值块(Value Block)执行增值操作(INCREMENT), 并将增值的块写(TRANSFER)到目标块. 写入的数据块地址不能为卡片的第一个数据块和密钥块. 即当扇区地址为 0x00 时,写入的块地址不能为 0x00 或 0x03. 当扇区地址在 0x01 至 0x1F 范围内时, 写入块地址不能为 0x03. 当扇区地址在 0x20 至 0x27 时, 写入块地址不能为 0x0F. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_PurseDecrement(uint8_t nChannel,uint8_t nSourceBlockAddress,uint8_t nDestinationBlockAddress,int32_t nValue) |
| 等效函数原型 | int uRfmif_DecrementTransfer (unsigned char bSrcBlock,unsigned char bDstBlock,unsigned char*bValue)<br>int sRfmif_DecrementTransfer (unsigned char bSrcBlock,unsigned char bDstBlock,unsigned char*bValue) |
| 参数 | nChannel 为射频通道号, 0-1 有效<br>nSourceBlockAddress 为源块地址,范围 0x00 至 0xFF.<br>nDestinationBlockAddress 为目标块地址,范围 0x00 至 0xFF. |

| | |
|---|---|
| | nValue 为从值块中减去的值 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该命令向 Mifare 系列卡的某一个值块(Value Block)执行减值操作(DECREMENT)，并将减值的块写(TRANSFER)到目标块. 写入的数据块地址不能为卡片的第一个数据块和密钥块. 即当扇区地址为 0x00 时,写入的块地址不能为 0x00 或 0x03. 当扇区地址在 0x01 至 0x1F 范围内时，写入块地址不能为 0x03. 当扇区地址在 0x20 至 0x27 时，写入块地址不能为 0x0F. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_PurseBackup(uint8_t nChannel,uint8_t nSourceBlockAddress,uint8_t nDestinationBlockAddress) |
| 等效函数原型 | int uRfmif_RestoreTransfer(unsigned char bSrcBlock,unsigned char bDstBlock)<br>int sRfmif_RestoreTransfer(unsigned char bSrcBlock,unsigned char bDstBlock) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nSourceBlockAddress 为源块地址,范围 0x00 至 0xFF.<br>nDestinationBlockAddress 为目标块地址,范围 0x00 至 0xFF. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该命令首先备份(RESTORE)Mifare 系列卡的某一个值块(Value Block)，并执行转移操作(TRANSFER),将备份的块写到别的块中. 写入的数据块地址不能为卡片的第一个数据块和密钥块. 即当扇区地址为 0x00 时,写入的块地址不能为 0x00 或 0x03. 当扇区地址在 0x01 至 0x1F 范围内时，写入块地址不能为 0x03. 当扇区地址在 0x20 至 0x27 时，写入块地址不能为 0x0F. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_AutoSleep(uint8_t nChannel,uint32_t nTimeout) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nTimeout 为超时时间设置,单位为毫秒. 为 0xFFFFFFFF 时,则禁用超时功能. 系统默认超时时间为 4000,即 4 秒. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数用于定时控制射频输出状态.当用户执行寻卡操作,打开射频后，定时器就会启动. 每次卡片操作(如发送 APDU 命令,读卡,三重认证操作等)都会重置定时器. 当用户在 4 秒钟内没有任何操作，且射频处在打开状态，程序就会自动关闭射频. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_SetEncryptionKeySet(uint8_t nChannel,uint8_t nKey) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nKey 如果为 0,表示使用 Philips 定义的标准密钥集,如果为 1,表示使用上海传输密钥集. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数设置射频调制芯片使用的传输密钥集. 例如,如果我们需要读写基于 FM11RF08SH 的非接触卡，需要使用上海传输密钥集.用户需要将 nKey 设置为 0x01，才能进行正常的三重认证.如果需要读写基于 MF1S50 的非接触卡，nKey 应设置为 0x00.<br>这个设置只需要在初始化时调用一次即可，系统默认使用 Philips 定义的标准密钥集.<br>注意:此函数需要调制芯片为 FM1715，对于 RC531 没有效果. 请参考 2.7 节. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_Request(uint8_t nChannel, uint8_t nRequestAll,uint8_t* pLength, uint8_t* pRequestData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nRequestAll 为寻卡模式,0 执行 REQB 命令，1 执行 WUPB 命令<br>pLength 为返回数据总长度<br>pRequestData 为返回数据，即 PUPI(4 字节) +AppData(4 个字节)+ ProtocolInfo(3 字节)，缓冲区长度至少为 11 字节 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数会查询某射频通道是否有 TypeB 卡存在，如果有卡，则该命令返回成功，用户通过卡片返回的 ATQB 等信息来判断下面的操作.该命令执行由 ISO14443-3 定义的 TypeB 寻卡命令的操作.在返回的数据中，PUPI 为卡片响应 4 字节伪 UID 数据，AppData 为卡片响应的 4 字节应用数据,ProtocolInfo 为卡片支持的协议类型. 这三个参数的定义请参考 ISO14443 标准. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_Attribute(uint8_t nChannel,uint8_t nParam3,uint8_t nExpectedDataRate,uint8_t* pCurrentDataRate,uint8_t* pINF,uint8_t* pINFLength) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nParam3 为 ATTRIB 命令的参数 3，对于 AT88RF 系列的卡片，该参数为 0x00，按照 ISO14443-3 标准，该参数为 0x01 |

| | |
|---|---|
| | nExpectedDataRate 为期望 TypeB 卡通信速率,其末尾 2 位(LSB,LSB+1)为 DRI，即从 RC531 到卡片的 bit 传输速率. 第 2-3 位(LSB+2,LSB+3)为 DSI，即从卡片到 RC531 的 bit 传输速率. 如果值为 00，速率为 1；01 速率为 2；10 速率为 4；11 速率为 8. 分别对应着 106Kbps，212Kbps，424Kbps 和 848Kbps 传输速率. 例如，0x05 表示期望上行和下行传输速率均为 212KBps.<br>pCurrentDataRate 为实际 TypeB 卡通信速率,格式同上. 如果该指针为 NULL，则不返回数据.<br>pINF 指向 ATTRIB 命令中发送的 INF 值，如果函数执行成功，指向 INF 响应数据<br>pINFLength 指向发送的 INF 值的长度，如果函数执行成功，指向 INF 响应数据的长度 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 在完成寻卡后，如果卡片属于 TypeB 卡，则应该调用该函数进行 TypeB 卡的归属操作.该函数主要执行由 ISO14443-3 标准定义的 ATTRIB 命令. 在该命令执行后，卡片就将进入 Active 状态.<br>**注意：** 参数 nExpectedDataRate 仅仅是期望使用的速率，TypeB 卡在 ProtocolInfo 中指示了当前卡支持的速率,读卡机会根据卡支持速率自动选择最接近的值.例如如果用户期望上行和下行传输速率均为 212KBps，但卡片只支持 106KBps,则读卡机会选择 106KBps 作为 ATTRIB 命令参数.参数 pCurrentDataRate 返回的值为 0x00. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_Read_Protection(uint8_t nChannel,uint8_t nUserArea,uint8_t nParameter,uint8_t nAddress,uint8_t nLength,uint8_t* pReceiveData) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nUserArea 为读取用户区命令，0 执行 Read System Zone，1 执行 Read User Zone 命令<br>nParameter 为命令参数<br>nAddress 为读取命令地址<br>nLength 为读取命令长度，读取的数据长度不能大于 16 字节<br>pReceiveData 为返回的数据 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数执行符合 ISO14443-3:2001 标准定义的 Read User Zone 和 Read System Zone 命令. 命令 Read User Zone 定义为 0x02，命令 Read System Zone 定义为 0x06，后面为三个字节的参数，分别是： |

| | |
|---|---|

| 操作 | nUserArea | nParameter | nAddress | nLength |
|---|---|---|---|---|
| 读取系统区 | 0 | 00 | addr | 数据长度 |
| 读取 Fuse 字节 | 0 | 01 | FF | 1 |
| 读取校验和 | 0 | 02 | FF | 2 |
| 读取用户区 | 1 | 00 | addr | 数据长度 |
| 读取用户区(大存储器) | 1 | addr high | addr low | 数据长度 |
| 读取带有 MAC 的用户区 | 1 | 80 | addr | 数据长度+2 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_Write_Protection(uint8_t nChannel,uint8_t nUserArea,uint8_t nParameter,uint8_t nAddress,uint8_t nLength,uint8_t* pTransmitData) |
| 参数 | nChannel 为射频通道号, 0-1 有效<br>nUserArea 为写入用户区命令, 0 执行 Write System Zone，1 执行 Write User Zone 命令<br>nParameter 为命令参数<br>nAddress 为写入数据地址<br>nLength 为写入数据长度，写入的数据长度不能大于 16 字节. 在防撕裂写入模式下，最大一次写入 8 字节.<br>pTransmitData 为写入的数据 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数执行符合 ISO14443-3:2001 标准定义的 Write User Zone 和 Write System Zone 命令. 命令 Write User Zone 定义为 0x03，命令 Write System Zone 定义为 0x04，后面为三个字节的参数分别是： |

| 操作 | nUserArea | nParameter | nAddress | nLength |
|---|---|---|---|---|
| 写入系统区 | 0 | 00 | addr | 数据长度 |
| 写入 Fuse 字节 | 0 | 01 | fuse addr | 1 |
| 写入带有 MAC 的系统区 | 0 | 80 | FF | 数据长度+2 |
| 写入用户区 | 1 | 00 | addr | 数据长度 |
| 写入用户区(大存储器) | 1 | addr high | addr low | 数据长度 |
| 写入带有 MAC 的用户区 | 1 | 80 | addr | 数据长度+2 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_SetUserZone(uint8_t nChannel,uint8_t nUserZone,uint8_t nAntiTearingWrite) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nUserZone 为设置的用户区索引，其范围从 0-15<br>nAntiTearingWrite 为执行防撕裂写入，即先写入临时 EEROM 中，再向实际区块写入数据 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数执行选择用户内存操作区域命令，用户区的数量随卡片不同，其范围从 0-15 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_VerifyCrypto(uint8_t nChannel,uint8_t nKeyIndex,uint8_t* pQ,uint8_t* pChallenge) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nKeyIndex 为密钥索引,0-3 为密钥种子,0x10-0x13 为段加密密钥<br>pQ 为主机生成的 8 字节随机数<br>pChallenge 为主机生成的 8 字节 Challenge |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数执行验证加密命令，参考 AT88RF 数据手册 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_SendChecksum(uint8_t nChannel,uint16_t nChecksum) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>nChecksum 为 2 字节的 MAC 值 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数执行发送校验和命令，参考 AT88RF 数据手册 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_Idle(uint8_t nChannel) |
| 参数 | nChannel 为射频通道号，0-1 有效 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数执行空闲命令，使卡进入空闲状态，清除当前密钥状态.<br>参考 AT88RF 数据手册 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RFID_TypeB_CheckPassword(uint8_t nChannel,uint8_t* pPassword) |
| 参数 | nChannel 为射频通道号，0-1 有效<br>pPassword 为 3 字节的密钥值 |
| 返回值 | 错误码(请参考 4.3 节) |

| | |
|---|---|
| 描述 | 该函数执行检测密钥命令，参考 AT88RF 数据手册 |

### 4.2.9 SAM 卡函数

SAM 卡函数可以独立操作 8 通道的 SAM 卡，其中包含两个函数族. 后缀为 Mapped 的函数考虑到兼容性，通道定义经过映射，SAM1 至 SAM8 分别对应卡槽索引 2,3,4,5,7,8,9,10,如下表所示. SAM 卡函数详细的使用方法，请参考 《SAM4A 编程手册》 第 3.6 节.

| 标准 SAM 卡函数 | 卡槽映射 SAM 卡函数 |
|---|---|
| SAM4A_SAM_Open<br>SAM4A_SAM_PowerOff<br>SAM4A_SAM_Reset<br>SAM4A_SAM_TranceiveAPDU<br>SAM4A_SAM_Close | SAM4A_SAM_OpenMapped<br>SAM4A_SAM_PowerOffMapped<br>SAM4A_SAM_ResetMapped<br>SAM4A_SAM_TranceiveAPDUMapped<br>SAM4A_SAM_CloseMapped |
| 卡槽索引 nSlot 范围为 0-7 | 卡槽索引 nMappedSlot 范围为 2,3,4,5,7,8,9,10 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SAM_OpenMapped (void) |
| 等效函数原型 | int OpenSimMoudle(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 初始化 SAM 卡模块,并设置卡槽映射. |

| | |
|---|---|
| 函数原型 | void SAM4A_SAM_PowerOffMapped (uint8_t nMappedSlot) |
| 等效函数原型 | void IccPowerOff (unsigned char slot) |
| 参数 | nMappedSlot 为映射卡槽索引 |
| 返回值 | 无 |
| 描述 | 关闭指定 SAM 卡电源. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SAM_ResetMapped (uint8_t nMappedSlot, uint32_t nBaudrate, uint8_t nVoltage, uint8_t* pLength, uint8_t* pATR,uint32_t nMode) |
| 等效函数原型 | int IccSimReset(unsigned int CardSelect,unsigned int uiRate,unsigned char ucVoltage,unsigned char* rLen,unsigned char* ATR,unsigned int mode) |
| 参数 | nMappedSlot 为映射卡槽索引<br>nBaudrate 为 SAM 卡通信波特率，允许的值为 9600,19200,38400,56000,115200,230400 |

| | |
|---|---|
| | nVoltage 为 SAM 电源电压,允许的值为 18(1.8V), 3(3.3V),5(5V)<br>pLength 为 ATR 的实际长度<br>pATR 为复位应答数据的指针<br>nMode 为复位模式,为 0 时执行冷复位,为 1 时执行热复位.当 nMode 大于 1 时,将进行热复位并执行 PTS, 此时 nMode 表示 SAM 卡执行 PTS 使用的波特率. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 复位指定 SAM 卡, 获得其 ATR. 并执行 PTS, 设置通信的数据传输速率. 按照 ISO7816 协议, nBaudrate 参数默认为 9600. 但是国内大量使用的一些 SAM 卡,如建设部 SAM, 在复位时即为 38400 波特率. 用户需要根据不同的卡片设置该参数. nMode 大于 1 时,卡片将执行 PTS 命令, 调整通信使用的波特率的参数.例如,参数 nBaudrate=9600, nMode=38400, 表示 SAM 卡复位时初始波特率为 9600, 在复位后通过 PTS 协议设置通信波特率为 38400, 在后面的发送 APDU 命令时, 使用的就是 38400 波特率.<br>注意: 当前读卡机不支持 230400 波特率, 参数 230400 将被置为 115200. SAM 卡电源电压也不能被改变, 无论参数 nVoltage 为何值, 一直为 3.3V |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SAM_TranceiveAPDUMapped (uint8_t nMappedSlot ,uint8_t* pTransmitBuffer,uint32_t nTransmitLength,uint8_t* pReceiveBuffer,uint32_t* pReceiveLength,uint16_t* pSW) |
| 等效函数原型 | int Sim_Apdu(unsigned int Slot,unsigned char *buffer,unsigned int length,unsigned char *rbuffer,unsigned int *Revlen,unsigned short *SW) |
| 参数 | nMappedSlot 为映射卡槽索引<br>pTransmitBuffer 为发送 APDU 数据包缓冲区指针<br>nTransmitLength 为发送 APDU 数据包长度<br>pReceiveBuffer 为接收 APDU 数据包缓冲区指针<br>pReceiveLength 为接收 APDU 数据包长度<br>pSW 为 APDU 命令状态字 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数向 SAM 卡发送 APDU 命令, 并接受卡的返回数据. SAM 卡的 APDU 命令由 ISO7816 标准定义, 并使用半双工字符传输模式 (T=0). SAM 卡命令缓冲区长度为 512 字节, 这意味着用户最大可以发送的 APDU 命令长度或返回 APDU 响应长度为 500 字节. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SAM_CloseMapped (uint8_t nMappedSlot) |
| 等效函数原型 | int CloseSimModule(unsigned int ucPlusinID) |
| 参数 | nMappedSlot 为映射卡槽索引 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 关闭 SAM 卡模块. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SAM_Open (void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 初始化 SAM 卡模块,并设置默认卡槽映射. |

| | |
|---|---|
| 函数原型 | void SAM4A_SAM_PowerOff (uint8_t nSlot) |
| 参数 | nSlot 为默认卡槽索引 |
| 返回值 | 无 |
| 描述 | 关闭指定 SAM 卡电源. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SAM_Reset (uint8_t nSlot, uint32_t nBaudrate, uint8_t nVoltage, uint8_t* pLength, uint8_t* pATR,uint32_t nPTSBaudrate) |
| 参数 | nSlot 为默认卡槽索引<br>nBaudrate 为 SAM 卡复位通信波特率, 允许的值为 9600,19200,38400,56000,115200<br>nVoltage 为 SAM 电源电压,允许的值为 18(1.8V), 3(3.3V),5(5V)<br>pLength 为 ATR 的实际长度<br>pATR 为复位应答数据的指针<br>nPTSBaudrate 为 SAM 卡 PTS 通信波特率, 允许的值为 0,9600,19200,38400,56000,115200 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 复位指定 SAM 卡, 获得其 ATR. 并执行 PTS, 设置通信的数据传输速率. 按照 ISO7816 协议, nBaudrate 参数默认为 9600. 但是国内大量使用的一些 SAM 卡,如建设部 SAM, 在复位时即为 38400 波特率. 用户需要根据不同的卡片设置该参数.例如,参数 nBaudrate=9600, nPTSBaudrate=38400, 表示 SAM 卡复位时初始波特率为 9600, 在复位后通过 PTS 协议设置通信波特率为 38400, 在后面的发送 APDU 命令时, 使用的就是 38400 波特率.<br>**注意：**当前读卡机不支持 230400 波特率, 参数 230400 将被置为 115200. SAM 卡电源电压也不能被改变, 无论参数 nVoltage 为何值, 一直为 3.3V |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SAM_TranceiveAPDU(uint8_t nSlot ,uint8_t* pTransmitBuffer,uint32_t nTransmitLength,uint8_t* pReceiveBuffer,uint32_t* pReceiveLength,uint16_t* pSW) |
| 参数 | nSlot 为默认卡槽索引<br>pTransmitBuffer 为发送 APDU 数据包缓冲区指针<br>nTransmitLength 为发送 APDU 数据包长度<br>pReceiveBuffer 为接收 APDU 数据包缓冲区指针<br>pReceiveLength 为接收 APDU 数据包长度<br>pSW 为 APDU 命令状态字 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 该函数向 SAM 卡发送 APDU 命令，并接受卡的返回数据. SAM 卡的 APDU 命令由 ISO7816 标准定义，并使用半双工字符传输模式(T=0). SAM 卡命令缓冲区长度为 512 字节，这意味着用户最大可以发送的 APDU 命令长度或返回 APDU 响应长度为 500 字节. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SAM_Close (uint8_t nSlot) |
| 参数 | nSlot 为默认卡槽索引 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 关闭 SAM 卡模块. |

### 4.2.10 RTC 函数

RTC 函数用于管理实时时钟(Real Time Clock)和 Linux 系统时钟之间的同步. RTC 电路在系统断电时，可以由锂电池继续供电. 在由系统供电时，则可以为锂电池充电，并保证 RTC 电路始终处在工作状态. **注意：** RTC 需要锂电池，电池型号为 LIR1220，电压 3.6V

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RTC_GetDate(uint8_t* pBCDTime) |
| 等效函数原型 | int GetDateTime(unsigned char*Time) |
| 参数 | pBCDTime 为读取到的当前时间，格式为 YYYYMMDDHHMMSS |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 从 Linux 系统时钟中读取当前时间. 当 RTC 计时值和 Linux 时间不匹配时，以 RTC 为准. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_RTC_SetDate(uint8_t* pBCDTime) |
| 等效函数原型 | int SetDateTime(unsigned char*Time) |
| 参数 | pBCDTime 为设置的当前时间，格式为 YYYYMMDDHHMMSS |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 设置处理器 RTC 和 Linux 系统的时钟. |

| | |
|---|---|
| 函数原型 | time_t SAM4A_RTC_GetTime(time_t* pTime) |
| 等效函数原型 | time_t time (time_t* timer) |
| 参数 | pTime 为读取到的当前时间,为标准时间点（epoch）到现在的时间经过的秒数. |
| 返回值 | 读取到的当前时间 |
| 描述 | 从 Linux 系统时钟中读取当前时间. 当 RTC 计时值和 Linux 时间不匹配时, 以 RTC 为准. |

### 4.2.11 铁电存储器函数

铁电存储器作为一种快速的非易失存储器(NVRAM), 可以用于保存应用程序的关键性数据,以保证出现系统掉电时, 应用程序可以恢复之前的工作状态. SAM4A 读卡机的 FERAM 尺寸为 128K 字节, 其中前 64K 字节为厂商保留空间, 用于存储系统引导程序. 用户可以自由使用后 64K 字节存储空间.

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_FERAM_Open(void) |
| 等效函数原型 | int FeromOpen(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 初始化 FERAM 模块 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_FERAM_Close(void) |
| 等效函数原型 | int FeromClose(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 关闭 FERAM 模块 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_FERAM_Write(uint32_t nDestinationAddress,uint32_t nWriteLength,uint8_t* pWriteBuffer) |
| 等效函数原型 | int FeSysWrite(unsigned int lFeAddr,unsigned int lrLength,unsigned char* pRtr) |
| 参数 | nDestinationAddress 为写入 FERAM 的地址,范围为 0x0-0xFFFF<br>nWriteLength 为写入 FERAM 的数据长度, 最大不能超过 512 字节<br>pWriteBuffer 为写入 FERAM 的数据缓冲区指针<br>**注意：** 缓冲区长度至少比写入 FERAM 的数据长度大 10 字节, 且在调用过该函数后, pWriteBuffer 的内容将被改变. |
| 返回值 | 错误码(请参考 4.3 节) |

| | |
|---|---|
| 描述 | 将缓冲区的数据写入 FERAM，如果该段地址被保护，数据将不会被写入存储器. 如果写入数据地址超出了 0xFFFF，将返回错误. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_FERAM_Read(uint32_t nSourceAddress,uint32_t nReadLength,uint8_t* pReadBuffer) |
| 等效函数原型 | int FeSysRead(unsigned int lFeAddr, unsigned int lrLength, unsigned char* pRtr) |
| 参数 | nSourceAddress 为读取 FERAM 的地址,范围为 0x0-0xFFFF<br>nReadLength 为读取 FERAM 的数据长度，最大不能超过 512 字节<br>pReadBuffer 为读取 FERAM 的数据缓冲区指针<br>**注意：**缓冲区长度至少比读取 FERAM 的数据长度大 10 字节 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 从 FERAM 中读取数据，如果读取数据地址超出了 0x3FFF,将返回错误. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_FERAM_SetProtectStatus(uint8_t nProtectMode) |
| 参数 | nProtectMode 为保护模式，0 为无保护，1 为保护地址 0x8000-0xFFFF，2 为保护地址 0x0000-0xFFFF |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 设置 FERAM 存储器的数据保护区，被设置保护的区域无法写入，但是可以读出数据. |

| | |
|---|---|
| 函数原型 | uint8_t SAM4A_FERAM_GetProtectStatus (void) |
| 参数 | 无 |
| 返回值 | 返回保护模式，0 为无保护，1 为保护地址 0x8000-0xFFFF，2 为保护地址 0x0000-0xFFFF |
| 描述 | 读取 FERAM 存储器的数据保护设置 |

### 4.2.12 LED 函数

LED 函数用于控制三个用户可编程状态指示灯和 1 个 4 位 8 段数码管.

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_LED_Open(void) |
| 等效函数原型 | int Led_DisOpen(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 初始化 LED 外设芯片，所有 LED 均处在熄灭状态. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_LED_Close(void) |
| 等效函数原型 | int Led_DisClose(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 关闭 LED 外设芯片. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_LED_Printf(uint8_t nDigitalMask,const char* pFormatString, ...) |
| 等效函数原型 | void Led_printf(IN MODE_FLICK mode,char const *fmt, ...) |
| 参数 | nDigitalMask 为数码管显示的掩码, 4 个数码管分别用 0x01,0x02,0x04,0x08 表示.<br>pFormatString 为格式字符串. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 控制 8 段数码管显示指定的信息, 有效的显示字符为 0 至 9, a 至 f,减法符号,下划线和小数点. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_LED_TurnOn(uint8_t nLEDMask) |
| 参数 | nLEDMask 为 LED 显示的掩码, 3 个 LED 管分别用 0x01,0x02,0x04 表示. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 点亮对应的 LED. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_LED_TurnOff(uint8_t nLEDMask) |
| 参数 | nLEDMask 为 LED 显示的掩码, 3 个 LED 管分别用 0x01,0x02,0x04 表示. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 熄灭对应的 LED. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_LED_Blink(uint8_t nLEDMask) |
| 参数 | nLEDMask 为 LED 显示的掩码, 3 个 LED 管分别用 0x01,0x02,0x04 表示. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 使对应的 LED 闪烁. |

### 4.2.13 GPIO 函数

GPIO 函数用于控制两个用户可编程引脚的状态. 这两个引脚连接到串口 2 的 pin1 和 pin6, 插座编号为 J11.

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_GPIO_SetDirection(uint8_t nGPIOMask,uint8_t nGPIODirection) |
| 参数 | nGPIOMask 为 GPIO 的掩码,2 个引脚分别用 0x01,0x02 表示.<br>nGPIODirection 为 GPIO 的方向掩码, 0 表示输入, 1 表示输出. |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 控制 GPIO 引脚的输入输出状态, 默认所有引脚均处在输入状态. 例如, SAM4A_GPIO_SetDirection(0x03,0x01)将 GPIO1 引脚置为输出,GPIO2 引脚置为输入. |

| | |
|---|---|
| 函数原型 | uint8_t SAM4A_GPIO_GetDirection(void) |
| 参数 | 无 |
| 返回值 | GPIO 引脚的输入输出状态 |
| 描述 | 获取 GPIO 引脚的输入输出状态. 默认所有引脚均处在输入状态, 即返回 0x03 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_GPIO_SetStatus(uint8_t nGPIOMask,uint8_t nGPIOStatus) |
| 参数 | nGPIOMask 为 GPIO 的掩码,2 个引脚分别用 0x01,0x02 表示.<br>nGPIOStatus 为 GPIO 的电平状态, 1 为高电平, 0 为低电平 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 设置 GPIO 引脚的输出状态, 该函数对输入引脚无效. |

| | |
|---|---|
| 函数原型 | uint8_t SAM4A_GPIO_GetStatus(uint8_t nGPIOMask) |
| 参数 | nGPIOMask 为 GPIO 的掩码,2 个引脚分别用 0x01,0x02 表示. |
| 返回值 | GPIO 引脚的电平状态 |
| 描述 | 获得 GPIO 引脚的电平输入状态, 该函数对输出引脚也有效. |

### 4.2.14 SD 卡函数

SD 卡函数用于检测 SD 卡插入状态并挂载 SD 卡, 本质上这是在调用 Linux 系统的 mount 命令. 因此, 建议用户在编程时使用 Linux 的挂载函数, 使得代码具有可移植性.

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SD_Check(void) |
| 等效函数原型 | int SDcard_Chk(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 检测 SD 是否被插入. **注意：** 当读卡机使用 HDMI 接口时,将在硬件上占用 SD 卡的检测引脚. 此时该函数始终返回未插入状态,用户应直接调用 SAM4A_SD_Mount 挂载 SD 卡. |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_SD_Mount(uint32_t nFilesystemType, uint32_t nMountFlag) |
| 等效函数原型 | int SDcard_mount(unsigned int FileType,unsigned int Sec) |
| 参数 | nFilesystemType 为挂载的文件系统类型,0 为自动探测文件系统,1 为 fat16 文件系统,2 为 fat32 文件系统,3 为 ntfs 文件系统,4 为 EXT2,5 为 EXT3.<br>nMountFlag 为挂载标志位,默认为 0 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 挂载 SD 卡. 该函数将调用定义于 sys/mount.h 的 mount 函数, 挂载源定义为/dev/mmcblk0p1, 挂载目标定义为/media/sd. 这相当于用户在 shell 中调用 mount /dev/mmcblk0p1 /media/sd |

### 4.2.15 看门狗函数

SAM4A 读卡机处理器内部带有看门狗, 默认处在关闭状态. 当看门狗启动时, 如果应用程序在一段时间内没有进行喂狗操作, 将触发系统的软复位. **注意：从 1.03 版开始, 看门狗的功能将由 Linux 内部看门狗驱动完成. 原驱动库的接口保留, 改为空内联函数.**

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Watchdog_Open(void) |
| 等效函数原型 | int WatchDog_Open(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 初始化看门狗, 默认复位时间为 20 秒 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Watchdog_Close(void) |
| 等效函数原型 | int WatchDog_Close(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 关闭看门狗 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Watchdog_SetTimeout(uint32_t nTimeout) |
| 等效函数原型 | int WatchDog_TimeOutValue(unsigned int lTimeOut) |
| 参数 | nTimeout 为超时计数，单位为秒 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 设置看门狗超时时间 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Watchdog_Feed(void) |
| 等效函数原型 | int clr_dog(void) |
| 参数 | 无 |
| 返回值 | 错误码(请参考 4.3 节) |
| 描述 | 喂看门狗 |

### 4.2.16 日志函数

日志函数用于将字符串型数据写入日志文件 SAM4A_LastLog.log ,日志文件默认存储在用户的 home 文件夹下的 log 目录中，即~/log 中. 当存储的日志文件尺寸大于限制时,日志函数将重命名为 SAM4A_XXX.log(XXX 为一个从 1 开始的数字)并创建一个新的日志文件. 每个日志文件都有一个内容完全相同的备份文件，备份文件的类型为 bak.

| | |
|---|---|
| 函数原型 | void SAM4A_Log_Setup(char* lpszLogDirectory,char* lpszLogSubdirectory,char* lpszLogName,uint32_t nMaximumLogSize) |
| 等效函数原型 | void SetAppLogMsg(char *LogMainPath,char *LogSubPath,char *LogNameFile,unsigned int AppLogSize) |
| 参数 | lpszLogDirectory 日志文件存储路径，默认为~<br>lpszLogSubdirectory 日志文件存储子目录，默认为/log<br>lpszLogName 日志文件名，默认为 SAM4A_LastLog.log<br>nMaximumLogSize 日志文件最大尺寸，默认为 1024K |
| 返回值 | 无 |
| 描述 | 设置日志函数的参数 |

| | |
|---|---|
| 函数原型 | void SAM4A_Log_Record(const char* lpszLogTitle,uint32_t nLogCategory, const char* lpszLogString) |
| 等效函数原型 | void AppLogOut(const char *pszTaskName, int iLogLevel, const char *pszFormat) |
| 参数 | lpszLogTitle 日志标题<br>nLogCategory 日志类型，0 为消息，1 为警告，2 为错误 |

| | |
|---|---|
| | lpszLogString 日志字符串 |
| 返回值 | 无 |
| 描述 | 向日志中写入字符串的函数 |

### 4.2.17 硬件参数函数

在加载接口库的时候，将自动读取当前的硬件信息，如处理器型号，硬件和软件版本号等. 用户可以通过结构变量 gSAM4A_HardwareInfo 获取这些信息.

| | |
|---|---|
| 函数原型 | unsigned char SAM4A_IsHardwareValid() |
| 参数 | 无 |
| 返回值 | 判断硬件信息是否有效，返回 1 为有效 |

| | |
|---|---|
| 函数原型 | SAM4A_HardwareInfo* SAM4A_GetHardwareInfo() |
| 参数 | 无 |
| 返回值 | 返回 SAM4A 读卡机的硬件参数，如果参数无效，则返回空指针 |

| 结构 SAM4A_HardwareInfo | | |
|---|---|---|
| 类型 | 变量 | 描述 |
| unsigned char[4] | m_uarrMagicNumber | 写入 FERAM 的数据校验<br>当第 4 个字节为 0xAA 时，硬件参数后面的值有效 |
| char [16] | m_carrCPUArchitecture | 处理器架构 |
| char [8] | m_carrCPUID | 处理器型号 |
| char [24] | m_carrBoardName | 主板名称 |
| char [8] | m_carrPCBVersion | 主板 PCB 版本号 |
| char [8] | m_carrCorePCBVersion | 主板核心板 PCB 版本号 |
| char [16] | m_carrManufactureDate | 生产日期 |
| unsigned char | m_uBoostMPUFrequency | 提升 MPU 频率<br>该参数如果为 1，则会提高开关电源的输出电压到 1.325V 并将处理器的工作频率设置为最大.<br>注意：如果 Linux 内核的 CPU 频率调节模块被安装，该参数无效 |
| unsigned char | m_uEraseLinux | 擦除 Linux 系统<br>如果该参数为 1，则 Uboot 会在启动时检查 Erase 按钮是否按下. 如果按下，则将 Linux 镜像的 NAND Flash 擦除。 |
| char [24] | m_carrUBootVersion | Uboot 版本号 |

| | | |
|---|---|---|
| char [16] | m_carrUBootCompileDate | Uboot 编译时间 |
| unsigned char [20] | m_uarrBoardUID | 主板 UID 号,确保每一个读卡机有独立的 ID |
| unsigned char [2] | m_uarrHFModulator | 调制芯片型号<br>0x00 MFRC500 ISO14443 TypeA<br>0x01 MFRC531 ISO14443 TypeA/B<br>0x02 FM1715 ISO14443 TypeA<br>0x03 MFRC663 ISO14443 TypeA/B/C<br>0x04 Unknown |
| char [8] | m_carrDriverVersion | 当前驱动版本号 |
| unsigned int | m_uFERAMSize | 用户可访问的 FERAM 的最大尺寸 |

### 4.2.18 电源管理函数

电源管理函数用于控制处理器的工作频率和电源电压.处理器默认使用 conservative 模式，即在处理器空闲时，降低处理器频率. 在负荷较大时增加处理器频率。

**警告：**处理器长时间工作在高频率可能导致温度过高，致使处理器寿命缩短. 如无特殊情况，请使用默认模式。

| 枚举 CPUFrequencyMode | | |
|---|---|---|
| 变量 | 值 | 描述 |
| OnDemand | 0x01 | 根据处理器负载调节频率, 倾向于使用高频率 |
| Conservative | 0x02 | 根据处理器负载调节频率, 倾向于使用低频率 |
| Performance | 0x04 | 一直使用最高频率 |
| PowerSave | 0x08 | 一直使用最低频率 |
| Userspace | 0x10 | 用户自行设置频率 |
| Unknown | 0x80 | 无效的频率模式 |

| | |
|---|---|
| 函数原型 | CPUFrequencyMode SAM4A_Power_GetCPUFrequencyMode(void) |
| 参数 | 无 |
| 返回值 | 处理器模式枚举 |
| 描述 | 获取当前处理器电源管理模式 |

| | |
|---|---|
| 函数原型 | uint32_t<br>SAM4A_Power_SetCPUFrequencyMode(CPUFrequencyMode eMode) |
| 参数 | eMode 为处理器模式枚举 |
| 返回值 | 系统错误码(请参考 4.3 节) |
| 描述 | 设置当前处理器电源管理模式 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Power_GetCPUFrequency(void) |
| 参数 | 无 |
| 返回值 | 处理器工作频率，以 KHz 为单位 |
| 描述 | 获取当前处理器工作的频率 |

| | |
|---|---|
| 函数原型 | uint32_t SAM4A_Power_SetCPUFrequency(uint32_t nCPUFrequency) |
| 参数 | nCPUFrequency 为处理器工作频率，以 KHz 为单位. 允许的值为 300000，600000，720000，800000，1000000 |
| 返回值 | 系统错误码(请参考 4.3 节) |
| 描述 | 设置当前处理器工作的频率,在调用该函数时，管理模式必须为 Userspace |

## 4.3 应用程序接口返回码描述

应用程序接口返回码类型为 4 字节无符号整型，返回 0x00000000 则表示无错误. 返回码的结构如下表所示:

| 31-24 位 | 23-16 位 | 15-8 位 | 7-0 位 |
|---|---|---|---|
| 保留(0x00) | 错误源<br>0x01 内核驱动出错<br>0x02 应用接口出错 | 错误硬件类型，见下表 | 错误码 |

| 编码 | 错误硬件类型 | 编码 | 错误硬件类型 |
|---|---|---|---|
| 0x00 | 无错误 | 0x09 | GPIO |
| 0x01 | LED | 0x0A | SPI 驱动 |
| 0x02 | HF 射频 | 0x0B | 看门狗 |
| 0x03 | 串口 | 0x0C | 日志函数 |
| 0x04 | 网络 | 0x0D | SD 卡 |
| 0x05 | RTC | 0x0E | 加密和校验函数 |
| 0x06 | SAM 卡 | 0x0F | UHF 射频 |
| 0x07 | 按键和蜂鸣器 | 0x10 | HDMI 接口 |
| 0x08 | 铁电存储器 | 0x11 | 电源管理 |

- LED 错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 非法的 LED 掩码参数 |
| 0x02 | 7 段数码管不支持的字符 |

- GPIO 错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 非法的 GPIO 掩码参数 |

- HF 射频错误码

| 编码 | 宏定义 | 错误描述 |
|---|---|---|
| 0x01 | SAM4A_HF_CollisionError | 传输位冲突 |
| 0x02 | SAM4A_HF_ParityError | 传输奇偶校验错 |
| 0x04 | SAM4A_HF_FramingError | 传输帧错误 |
| 0x08 | SAM4A_HF_CRCError | 传输 CRC 校验错误 |
| 0x10 | SAM4A_HF_FIFOOverflow | FIFO 溢出 |
| 0x20 | SAM4A_HF_AccessError | 寄存器访问错误 |
| 0x40 | SAM4A_HF_KeyError | 非法的密钥,三重认证错误 |
| 0x80 | SAM4A_HF_EEROMError | EEROM 操作错误 |
| 0x81 | SAM4A_HF_LoadKeyE2Error | 未能载入 EEROM 中密钥 |
| 0x82 | SAM4A_HF_RequestError | 寻卡操作失败 |
| 0x83 | SAM4A_HF_AntiCollisionError | 防重叠操作失败 |
| 0x84 | SAM4A_HF_UIDChecksumError | 卡片返回 UID 校验失败 |
| 0x85 | SAM4A_HF_SelectTagError | 选卡失败 |
| 0x86 | SAM4A_HF_AuthenError | 三重认证失败 |
| 0x87 | SAM4A_HF_AuthenTimeout | 认证超时 |
| 0x88 | SAM4A_HF_ReadCardError | 读卡错误 |
| 0x89 | SAM4A_HF_ReadByteCountError | 读卡返回数据长度错误 |
| 0x8A | SAM4A_HF_WriteCardError | 写卡错误 |
| 0x8B | SAM4A_HF_HaltError | 中止错误 |
| 0x8C | SAM4A_HF_IncrementError | 值块增错误 |
| 0x8D | SAM4A_HF_DecrementError | 值块减错误 |
| 0x8E | SAM4A_HF_RestoreError | 值块恢复错误 |
| 0x8F | SAM4A_HF_TransferError | 值块传输错误 |
| 0x90 | SAM4A_HF_RequestATSError | CPU 卡 RATS 命令失败 |
| 0x91 | SAM4A_HF_RATSCountError | ATS 返回数据长度错误 |

| | | |
|---|---|---|
| 0x92 | SAM4A_HF_RATSCRCError | ATS 返回数据 CRC 错误 |
| 0x93 | SAM4A_HF_RequestPPSError | PPS 命令失败 |
| 0x94 | SAM4A_HF_TranceiveError | APDU 收发包失败 |
| 0x95 | SAM4A_HF_TranceiveACKError | APDU 收发卡片没有返回 ACK |
| 0x96 | SAM4A_HF_TranceiveReceiveError | APDU 收发卡片接收错误 |
| 0x97 | SAM4A_HF_TranceiveBlockNumberError | APDU 收发块号错误 |
| 0x98 | SAM4A_HF_DeselectError | Deselect 命令失败 |
| 0x99 | SAM4A_HF_DeselectRespondError | Deselect 命令卡片没有应答 |
| 0x9A | SAM4A_HF_RequestAntiCollisionError | 防重叠环错误 |
| 0x9B | SAM4A_HF_SelectTagSAKError | 卡片返回无效 SAK |
| 0x9C | SAM4A_HF_ReceivePackageLengthError | 接收数据包长度错误 |
| 0x9D | SAM4A_HF_TransmitPackageLengthError | 发送数据包长度错误 |

- 串口错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 未知串口错误 |
| 0x02 | 通讯端口号为空 |
| 0x03 | 解析参数格式不对 |
| 0x04 | 设置参数出错 |
| 0x05 | 申请空间不足 |
| 0x06 | 无效的句柄 |
| 0x07 | 通讯端口不符 |
| 0x08 | 当前客户机超过最大连接个数 |
| 0x09 | 通讯超时 |

- 网络错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 非法的设备路径 |
| 0x02 | 参考参数格式不对 |
| 0x03 | 设置参数出错 |
| 0x04 | 申请Socket非法 |
| 0x05 | 通讯端口不符 |
| 0x06 | 当前客户机超过最大连接个数 |
| 0x07 | 通讯超时 |
| 0x08 | 读通讯端口数据出错 |
| 0x09 | 通讯端口中断 |
| 0x0A | 写端口数据出错 |

- RTC 错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 非法的时间格式 |
| 0x02 | 设置 RTC 失败 |
| 0x03 | 设置 Linux 时间失败 |

- SAM 卡错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 非法的卡槽索引 |
| 0x02 | 无效的波特率 |
| 0x03 | SAM 卡复位失败 |
| 0x04 | SAM 卡 APDU 命令失败 |

- 铁电存储器错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 写入数据地址超出范围 |
| 0x02 | 读取数据地址超出范围 |

- SPI 驱动错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 缓冲区溢出 |
| 0x02 | 非法的命令 |
| 0x03 | 非法的片选掩码 |

- 看门狗错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 无效的超时值 |
| 0x02 | 看门狗初始化失败 |
| 0x03 | 喂狗失败 |

- 日志函数错误码

| 编码 | 错误描述 |
|---|---|
| 0x01 | 不能打开日志文件 |
| 0x02 | 不能保存日志文件 |
| 0x03 | 日志设置文件异常 |

- 电源管理函数错误码

| 编码 | 错误描述 |
| --- | --- |
| 0x01 | sys 接口错误 |
| 0x02 | 频率错误 |

# 5. 读卡机选型表

| 型号 | 类型 | 应用 | 处理器 | 处理器速度 | 操作系统 | 射频通道 | SAM 卡 |
|---|---|---|---|---|---|---|---|
| ZC681 | 读卡机模块 | 模块内置射频功放，易于使用 | 无 | | 无 | 1 | 无 |
| TJRF | 读卡机模块 | | 无 | | 无 | 1 | 无 |
| SAM8 | 小读卡机 | 低成本读卡机 | LPC2214(ARM7) | 60MHz | RTOS | 1 | 8 个 SAM 卡 |
| SAM82 | 小读卡机 | 连接型读卡机 | STM32F217(ARM-CortexM3) | 120MHz | uCLinux | 2 | 8 个 SAM 卡 |
| SAM83 基础型 | 小读卡机 | 低成本读卡机 | STM32F217(ARM-CortexM3) | 120MHz | RTOS | 1 | 8 个 SAM 卡 |
| SAM83 扩展型 | 大读卡机 | 连接型读卡机 | STM32F217(ARM-CortexM3)<br>+NUC120(ARM-CortexM0) | 120MHz | uCLinux | 2 or 3 | 8 个 SAM 卡 |
| RFID8 | 小读卡机 | 多射频通道读卡机 | NUC120 (ARM-CortexM0) | 50MHz | RTOS | 8 | 8 个 SAM 卡 |
| RFID61 | 小读卡机 | 低成本读卡机 | LPC2214(ARM7) | 60MHz | RTOS | 1 | 4 个 SAM 卡 |
| RFID2 | 小读卡机 | 低成本读卡机 | LPC2214(ARM7) | 60MHz | RTOS | 2 | 8 个 SAM 卡 |
| SAM9260 | 大读卡机 | 连接型读卡机 | AT91SAM9260(ARM9)<br>+NUC120(ARM-CortexM0) | 210MHz | Linux or WinCE | 2 or 3 | 8 个 SAM 卡 |
| SAM4A | 大读卡机 | 高性能读卡机 | AM3352(ARM-CortexA8) | 1GHz | Linux or WinCE or Android | 2 | 8 个 SAM 卡 |

| Flash | RAM | 串口 | USB | 以太网 | SD 卡 | 视频/音频 |
|---|---|---|---|---|---|---|
| 无 | 无 | 0 | 0 | 无 | 无 | 无 |
| 无 | 无 | 0 | 0 | 无 | 无 | 无 |
| 内部 256KB<br>+可选外部 32Mbit | 内部 64KB<br>+可选外部 16Mbit | 2 | 0 | 无 | 无 | 无 |
| 内部 1024KB<br>+可选外部 64Mbit | 内部 128KB<br>+可选外部 16Mbit | 2 | 1 个 OTG 主机<br>或设备 | 10/100M | 有 | 无 |
| 内部 1024KB<br>+可选外部 64Mbit | 内部 128KB<br>+可选外部 16Mbit | 2 | 1 个 OTG 主机<br>或设备 | 无 | 无 | 无 |
| 内部 1024KB<br>+可选外部 512Mbit<br>+可选串行 512Mbit | 内部 128KB<br>+可选外部 32Mbit | 2 | 1 个 OTG 主机<br>或设备<br>+1 个 HID 设备 | 10/100M | 有 | 无 |
| 内部 64KB | 内部 8KB | 1 | 1 个 HID 设备 | 无 | 无 | 无 |
| 内部 256KB | 内部 64KB | 1 | 1 个 USB 串口 | 无 | 无 | 无 |
| 内部 256KB<br>+可选外部 32Mbit | 内部 64KB<br>+可选外部 16Mbit | 1 | 0 | 无 | 无 | 无 |
| 外部 256Mbit,最大<br>扩展到 2Gbit<br>+可选串行 1Gbit | 内部 8KB<br>+外部 256Mbit,<br>最大扩展到 512Mbit | 2 | 1 个主机<br>+ 1 个设备<br>+1 个 HID 设备 | 10/100M | 有 | 无 |
| 外部 8GBytes,最大<br>扩展到 32GBytes | 内部 128KB<br>+外部 1GBytes | 3 | 1 个主机<br>+ 1 个设备 | 10/100M | 有 | 有 |