安定宝协议

*更新记录*

*2019年7月31日 新增概述与附录crc16算法*

*2019年8月2日 补充示例*

*2019年8月5日 补充概述*

*2019年9月9日 补充xdata段说明*

# 概述

网络模块与接警中心之间使用安定宝协议DC05，DC07，DC09，并根据我司需要添加了部分事件代码。

网络模块开机后主动连接接警中心，接警中心的IP地址与端口号由网络模块配置工具进行配置。连接成功后网络模块报告一次主机状态，然后开始定时发送心跳包(NULL)，接收方应及时回应相同seq的ACK包；主机有事件（状态改变、产生警情）发生时上报接警中心。

接警中心远程控制发送撤防命令时，密码放在xdata段内。

本端收到除DUH/ACK包之外的其他有效数据包时，都应回应相同seq的ACK包，否则对端会进行重发。

# 传输

使用ASCII码传输

# 协议

## 1.格式（以下出现的< >仅作分隔符，实际数据中不存在）：

<LF><crc16><length><"id"><seq><Rrcvr><Lpref><#acct><data><\*xdata><timestamp><CR>

## 2.各字段说明：

### <LF>

固定起始符 0x0A

### <crc16>

固定长度4Bytes, 表示从id的左引号开始，至timestamp的最后一个字节结束(长度为<length>)计算出的crc16值

### <length>

从id的左引号开始，至timestamp的最后一个字节结束的字符串长度，以16进制数字的ASCII码表示，定长4Bytes

### <”id”>

事件类型，分为NULL, ACK, DUH, ADM-CID, HENG-BO等，分别为心跳、回应、错误、主机事件、恒博事件。除事件ADM-CID/HENG-BO外，其他三种类型的data段都为空值 [ ]。

### <seq>

序号，定长4 Bytes，代表的十进制数字为主机发送命令字的序号，范围0001~9999，超过9999则从1开始再次循环。seq被用来判断主机发出的ADM-CID/HENG-BO类型命令字是否得到接警平台的正确处理。例：平台要对主机发出的NULL/ADM-CID/HENG-BO类型事件进行回应，回应的seq以接收到的seq为准。主机收到回应后会判断seq与之前发送的seq是否相同：相同则说明该事件已得到处理，否则会进行重发。

### <Rrcvr>

以R开头的一段十六进制数字字符串，数字长度范围1~6，暂时仅占位，无意义。字段总长度2~7 Bytes。

### <Lpref>

以L开头的一段十六进制数字字符串，数字长度范围1~6，暂时仅占位，无意义。字段总长度2~7 Bytes。

### <#acct>

以#开头，后跟有限位十进制数字，表示主机的账号（有SIM卡的主机为手机号，无SIM卡的是出厂编号）

### <data>

格式为[]（左右中括号，无空格，定长2Byte）或[#acct|event gg zone]（有空格）。当事件类型为NULL/ACK/DUH时data段为[]。当事件类型为ADM-CID/HENG-BO时data段格式为[#acct|event gg zone]。

#### #acct

主机的安定宝ID，4位或6位或8位数字

#### event

定长4Bytes，事件代码，具体含义见后文。

#### gg

定长2 Bytes， 一般为00，当非00时有特殊含义，参考《安定宝协议扩展》

#### zone

防区号，范围0~999。zone为0表示主机自身事件，如布防、撤防、防拆等。1~999则为防区号，表示防区事件。定长3 Bytes。

### <\*xdata>

可选的字段，一般情况下留空，仅有限几种情况需要带xdata：撤防时使用xdata传输密码、索要防区信息时使用xdata做串口命令透传。具体格式参考《安定宝协议扩展》。

### <timestamp>

时间戳字段，格式为\_HH:MM:SS,mm-DD-YYYY，定长20Bytes

### <CR>

固定结束符，0x0D

# 事件码

## 主机状态

|  |  |
| --- | --- |
| 1400 | 撤防 |
| 3400 | 布防 |
| 1456 | 半布防 |
| 1120 | 紧急报警 |

## 防区报警

|  |  |
| --- | --- |
| 1130 | 盗警 |
| 1134 | 门铃 |
| 1110 | 火警 |
| 1121 | 胁迫 |
| 1151 | 煤气 |
| 1113 | 水警 |
| 1137 | 防拆 |

## 防区异常

|  |  |
| --- | --- |
| 1302 | 低电 |
| 3302 | 复电 |
| 1311 | 电池坏 |
| 1387 | 光扰 |
| 1381 | 失效 |
| 3381 | 恢复 |
| 1384 | 电源故障 |
| 3384 | 电源故障恢复 |
| 1380 | 其他故障 |
| 3380 | 其他故障恢复 |

## \*恒博私有事件码

|  |  |
| --- | --- |
| 1485 | 网络模块与主机485串口连接断开 |
| 3485 | 网络模块与主机485串口连接恢复 |
| 1701 | 撤防密码错误 |
| 1702 | 分机探头异常 |
| 3702 | 分机探头异常恢复 |
| 1703 | 分机电源异常 |
| 3703 | 分机电源异常恢复 |
| 1704 | 索要防区信息 |
| 2704 | 主机进入设置状态 |
| 3704 | 停止索要或拒绝索要 |
| 1705 | 查询分机信息 |
| 1706 | 写入主机信息 |
| 1707 | 标明主机类型为工程主机+网络模块 |
| 1709 | 手机用户SOS |
| 1711 | 手机用户消警 |
| 1717 | 标明主机类型为2050型GPRS主机 |
| 1799 | 主机信号强度变化 |

# 附录

## crc16算法

/\*\*

\* @brief 计算一个字节（char）的crc16值

\* @param c 要计算crc16的字节

\* @param crc 初始crc值

\* @return crc16

\*/

static unsigned short CalculateCRC\_char(char c, unsigned short crc)

{

static unsigned short crcTable[] = {

/\* DEFINE THE FIRST ORDER POLYINOMIAL TABLE \*/

0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,

0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,

0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,

0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,

0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81, 0x1A40,

0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,

0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,

0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,

0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,

0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,

0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,

0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,

0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,

0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,

0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,

0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,

0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,

0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,

0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,

0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,

0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,

0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,

0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,

0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,

0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,

0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,

0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,

0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,

0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,

0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,

0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,

0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040,

};

unsigned short CRC = crc;

auto tmp = static\_cast<unsigned short>(CRC >> 8) ^ (crcTable[static\_cast<unsigned char>(c) ^ static\_cast<unsigned char>(CRC & 0xFF)]);

CRC = static\_cast<unsigned short>(tmp);

return CRC;

}

/\*\*

\* @brief 计算一段数据包的crc16值

\* @param buff 要计算crc16的数据包指针

\* @param len 要计算crc16的数据包长度

\* @param crc 初始crc值

\* @return crc16

\*/

static unsigned short CalculateCRC(const char\* buff, size\_t len, unsigned short crc = 0)

{

unsigned short CRC = crc;

for (size\_t i = 0; i < len; ++i) {

CRC = CalculateCRC\_char(buff[i], CRC);

}

return CRC;

}

## 示例

### 主机000000布防

"\n02130047\"HENG-BO\"0000R1234L6789#000000[#000000|3400 00 000]\_17:15:29,12-09-2015\r"

### 主机000000心跳

"\n93B30031\"NULL\"0000R1234L6789#000000[]\_17:24:14,12-09-2015\r"

### 接警中心心跳回应

"\nF0550030\"ACK\"0000R1234L6789#000000[]\_17:27:09,12-09-2015\r"