HDLC-ATC 通告接口

1 概述.....2
1.1 目的.....2
1.2 适用产品.....2
1.3 字节顺序.....2
1.4 对齐方式.....2
1.5 基本数据类型.....2
2 利用 DMS 激活通告接口.....3
2.1 组网.....3
2.2 配置.....3
3 通告消息.....4
3.1 DMS 消息组成.....4
3.2 DMSG_HEADER：DMS 消息头.....4
3.2.1 字段定义.....4
3.2.2 结构定义.....4
3.2.3 设备类型编码.....5
3.2.4 消息类型.....5
3.3 消息接收注意事项.....5
3.4 设备信息通告：DMSG_SEARCH_ACK 消息.....6
3.4.1 错误代码定义.....6
3.4.2 数据结构.....7
3.4.3 字段描述.....7
3.5 统计报告通告：DMSG_REPORT_GET_ACK 消息.....8
3.5.1 定义.....8
3.5.2 数据结构.....8
3.5.3 字段描述.....9

1 概述

1.1 目的

亚册 HDLC-ATC-400、HDLC-ATC-800、HDLC-ATC-1600 提供了通告接口，能够主动发送通告数据，为监控服务器或计算机提供设备运行信息、数据通信收发统计报告。

1.2 适用产品

- HDLC-ATC-400
- HDLC-ATC-800
- HDLC-ATC-1600

本文统称 **HDLC-ATC**。

1.3 字节顺序

字节顺序是指占内存多于一个字节类型的数据在内存中的存放顺序，通常有：

- 小端字节序（Little-Endian）：指低字节数据存放在内存低地址处，高字节数据存放在内存高地址处；
- 大端字节序（Big-Endian）：指高字节数据存放在低地址处，低字节数据存放在高地址处。

由于用户系统多用基于小字节序的 x86 系列 CPU，因此 **HDLC-ATC 编程接口统一采用小端字节（Little-Endian）顺序定义**。

1.4 对齐方式

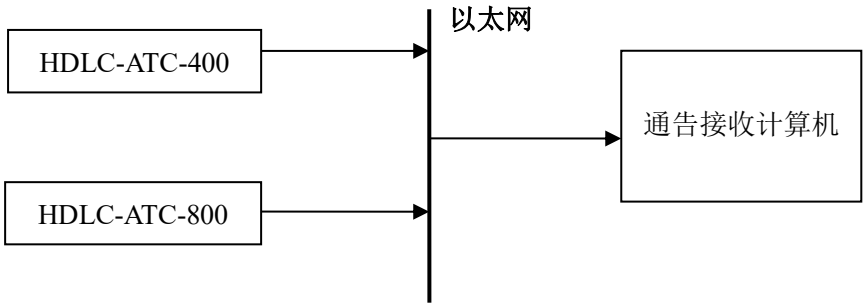
为了减少数据包长度，HDLC-ATC 编程接口数据结构采用 **1 字节对齐方式**。

1.5 基本数据类型

- **uint8_t**：unsigned char，1 字节无符号整型数
- **uint16_t**：unsigned short，2 字节无符号整型数
- **uint32_t**：unsigned word，4 字节无符号整型数

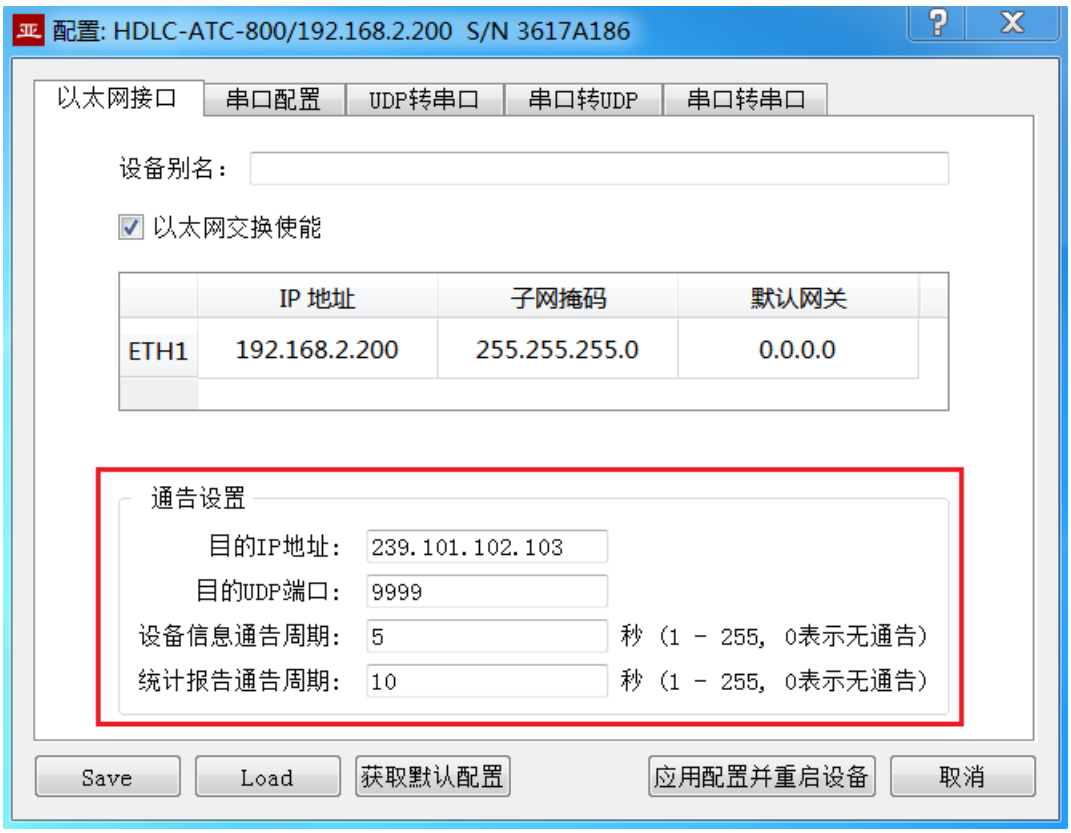
2 利用 DMS 激活通告接口

2.1 组网



2.2 配置

用户可以运行 DMS，进入“以太网接口”配置，进行通告设置。



通告设置包括如下参数：

参数	描述
目的 IP 地址	通告消息发送目的 IP，可以为单播或组播地址
目的 UDP 端口	通告采用 UDP 报文发送，目的 UDP 端口号可设
设备信息通告周期	设备运行信息通告的发送周期，1~255 秒
统计报告通告周期	收发统计报告的发送周期，1~255 秒

3 通告消息

3.1 DMS 消息组成

通告消息属于 DMS 消息，DMS 为设备与管理监控计算机之间的专用交互消息接口。

DMS 消息由 message header（消息头）和 message body（消息体）组成，其中消息体最大长度不能超过 1440：

消息头 Message header	消息体 Message body

3.2 DMSG_HEADER：DMS 消息头

3.2.1 字段定义

消息头字段按顺序定义如下：

字段	字节数	描述
dmsFlag	2	DMS 消息标志字段，必须为 0x444D
dmsVersion	1	DMS 消息版本号，必须为 0x20，表示 V2.0 版本
reserved	1	保留
sndDevType	4	发送方设备类型编码
sndDevSN	4	发送方设备序列号
rcvDevType	4	接收方设备类型编码，全 1 为所有类型设备
rcvDevSN	4	接收方设备序列号，全 1 为所有设备
msgType	2	消息类型
msgLen	2	消息长度（字节数），含消息头，不超过 1440

3.2.2 结构定义

```
struct DMSG_HEADER
{
    uint16_t   dmsFlag;        // DMS 消息标志
    uint8_t    dmsVersion;     // DMS 消息发送方 DMS 的版本号
    uint8_t    reserved;       // 保留

    uint32_t   sndDevType;     // 发送方设备类型编码
    uint32_t   sndDevSN;       // 发送方设备序列号

    uint32_t   rcvDevType;     // 接收方设备类型编码
    uint32_t   rcvDevSN;       // 接收方设备序列号

    uint16_t   msgType;        // 消息类型
    uint16_t   msgLen;         // 消息长度，含消息头
}
```

3.2.3 设备类型编码

设备类型编码	描述
0xFFFFFFFF	所有类型
0x00000711 0x00000720	HDLC-ATC400、HDLC-ATC-800、HDLC-ATC-1600
0x10000000	DMS

3.2.4 消息类型

消息名称	类型编码	描述
DMSG_SEARCH_ACK_ID	0x0011	设备信息通告、设备搜索响应
DMSG_REPORT_GET_ACK_ID	0x1202	统计报告通告、设备报告查询响应

宏定义如下：

```
#define DMSG_SEARCH_ACK_ID          0x0011
#define DMSG_REPORT_GET_ACK_ID      0x1202
```

3.3 消息接收注意事项

在未来的版本中，消息长度有可能增加，为了保证软件的稳定性，用户应该用 DMSG_BUFFER 大小的缓冲区接收该消息。

```
#define DMSG_MAXLEN                 1440

struct DMSG_BUFFER
{
    DMSG_HEADER header;
    char data[DMSG_MAXLEN];

};
```

3.4 设备信息通告：DMSG_SEARCH_ACK 消息

3.4.1 错误代码定义

错误代码采用位掩码的方式定义，全 0 表示系统正常。

```
#define YC_ER_HDLC_ATC_800_NONE       0                  // 0 表示无故障
#define YC_ER_HDLC_ATC_800_CLOCK      0x00000001UL       // 时钟故障
#define YC_ER_HDLC_ATC_800_RAM        0x00000002UL       // 内存故障
#define YC_ER_HDLC_ATC_800_XADC       0x00000004UL       // XADC 故障
#define YC_ER_HDLC_ATC_800_FLASH      0x00000008UL       // 闪存故障
#define YC_ER_HDLC_ATC_800_GPIO       0x00000010UL       // IO 接口故障
#define YC_ER_HDLC_ATC_800_SWITCH     0x00000020UL       // 以太网交换功能故障
#define YC_ER_HDLC_ATC_800_UART       0x00000040UL       // 异步串口故障
#define YC_ER_HDLC_ATC_800_FPGA       0x00000080UL       // FPGA 故障
#define YC_ER_HDLC_ATC_800_UDPCLT     0x00000100UL       // UDP 客户端异常
#define YC_ER_HDLC_ATC_800_UDPSVR     0x00000200UL       // UDP 服务器异常
#define YC_ER_HDLC_ATC_800_PLL        0x00000400UL       // 锁相环故障
#define YC_ER_HDLC_ATC_800_BAUDRATE   0x00000800UL       // 波特率异常
#define YC_ER_HDLC_ATC_800_DFTSET     0x04000000UL       // 以默认配置启动
#define YC_ER_HDLC_ATC_800_DNA        0x08000000UL       // 设备身份 ID 异常
#define YC_ER_HDLC_ATC_800_FACTORY    0x10000000UL       // 出厂配置异常
#define YC_ER_HDLC_ATC_800_CONFIG     0x20000000UL       // 设备配置异常
#define YC_ER_HDLC_ATC_800_HWVER      0x40000000UL       // 硬件版本号异常
#define YC_ER_HDLC_ATC_800_SN         0x80000000UL       // 序列号异常
```

3.4.2 数据结构

```
struct DMSG_SEARCH_ACK {
    DMSG_HEADER header;

    char deviceAlias[32];          // 设备别名
    uint32_t errorCodes;           // 错误代码，位组合

    uint32_t reserved;
    uint32_t firmwareVersion;      // 设备固件版本
    uint32_t fpgaVersion;          // FPGA 版本

    char reserved256[256];
};
```

3.4.3 字段描述

DMSG_SEARCH_ACK 消息字段按顺序定义为：

字段	类型	描述
header	DMSG_HEADER	消息头，DMSG_HEADER
deviceAlias [ ]	char [ ]	设备别名字符串
errorCodes	uint32_t	设备故障代码
firmwareVersion	uint32_t	固件版本
fpgaVersion	uint32_t	FPGA 版本

3.5 统计报告通告：DMSG_REPORT_GET_ACK 消息

3.5.1 定义

```
#define YC_HDLC_ATC_800_SER_NUM        16    // 设备串口数量
#define YC_HDLC_ATC_800_UDPCLT_NUM     16    // 设备支持 UDP client 数量
#define YC_HDLC_ATC_800_UDPSVR_NUM     16    // 设备支持 UDP Server 数量
```

3.5.2 数据结构

数据结构定义：

```
struct DMSG_REPORT_GET_ACK
{
    uint32_t runSeconds;    // 设备加电后运行秒数

    // DMS 统计
    uint32_t dms_tx_pkt;
    uint32_t dms_tx_fail;
    uint32_t dms_rx_pkt;
    uint32_t dms_rx_invalid;

    // 串口收发统计
    uint32_t ser_tx_pkt[YC_HDLC_ATC_800_SER_NUM];
    uint32_t ser_tx_overflow[YC_HDLC_ATC_800_SER_NUM];
    uint32_t ser_tx_toolong[YC_HDLC_ATC_800_SER_NUM];
    uint32_t ser_rx_pkt[YC_HDLC_ATC_800_SER_NUM];
    uint32_t ser_rx_crc_error[YC_HDLC_ATC_800_SER_NUM];
    uint32_t ser_rx_overflow[YC_HDLC_ATC_800_SER_NUM];
    uint32_t ser_rx_tooshort[YC_HDLC_ATC_800_SER_NUM];
    uint32_t ser_rx_toolong[YC_HDLC_ATC_800_SER_NUM];

    // UDP client
    uint32_t udp_tx_pkt[YC_HDLC_ATC_800_UDPCLT_NUM];
    uint32_t udp_tx_fail[YC_HDLC_ATC_800_UDPCLT_NUM];

    // UDP Server
    uint32_t udp_rx_pkt[YC_HDLC_ATC_800_UDPSVR_NUM];
    uint32_t udp_rx_fail[YC_HDLC_ATC_800_UDPSVR_NUM];

    // 内部保留
    uint32_t reserved1[64];
}
```

3.5.3 字段描述

DMSG_REPORT_GET_ACK 消息字段按顺序定义为：

字段	类型	描述
header	DMSG_HEADER	消息头，DMSG_HEADER
runSeconds	uint32_t	设备加电后运行秒数
dms_tx_pkt	uint32_t	设备发送 DMS 消息计数
dms_tx_fail	uint32_t	设备发送 DMS 消息失败计数
dms_rx_pkt	uint32_t	设备接收 DMS 消息计数
dms_rx_invalid	uint32_t	设备接收无效 DMS 消息计数
ser_tx_pkt [ ]	uint32_t [ ]	串口发帧计数
ser_tx_overflow [ ]	uint32_t [ ]	串口发送溢出计数
ser_tx_toolong [ ]	uint32_t [ ]	串口发送超长帧计数
ser_rx_pkt [ ]	uint32_t [ ]	串口收帧计数
ser_rx_crc_error [ ]	uint32_t [ ]	串口接收 CRC 错误计数
ser_rx_overflow [ ]	uint32_t [ ]	串口接收溢出计数
ser_rx_tooshort [ ]	uint32_t [ ]	串口接收超短帧计数
ser_rx_toolong [ ]	uint32_t [ ]	串口接收超长帧计数
udp_tx_pkt [ ]	uint32_t [ ]	UDP 发包计数
udp_tx_fail [ ]	uint32_t [ ]	UDP 发送错误计数
udp_rx_pkt [ ]	uint32_t [ ]	UDP 收包计数
udp_rx_fail [ ]	uint32_t [ ]	UDP 接收错误计数