SDLC-ETH 配置管理接口

1 概述 3
1.1 目的 3
1.2 适用产品 3
1.3 字节顺序 3
1.4 对齐方式 3
1.5 基本数据类型 3
2 DMS 接口 4
2.1 DMS 4
2.2 通信方式 4
2.3 DMS 消息 4
2.4 DMSG_HEADER：DMS 消息头 5
2.4.1 结构定义 5
2.4.2 字段定义 5
2.4.3 消息头填写建议 6
2.4.4 设备类型编码 6
2.4.5 消息类型 6
3 设备搜索及状态接口 8
3.1 功能描述 8
3.2 DMSG_SEARCH 消息 8
3.2.1 数据结构 8
3.2.2 字段描述 8
3.3 DMSG_SEARCH_ACK 消息 9
3.3.1 数据结构 9
3.3.2 字段描述 9
3.3.3 错误代码定义 9
4 设备报告查询接口 10
4.1 功能描述 10
4.2 DMSG_REPORT_GET 消息 10
4.2.1 数据结构 10
4.2.2 字段描述 10
4.3 DMSG_REPORT_GET_ACK 消息 11
4.3.1 定义 11
4.3.2 数据结构 11
4.3.3 字段描述 12
5 设备配置查询接口 14
5.1 功能描述 14
5.2 DMSG_CONFIG_GET 消息 14
5.2.1 数据结构 14
5.2.2 字段描述 14
5.3 DMSG_CONFIG_GET_ACK 消息 14

5.3.1 定义.....14
5.3.2 数据结构.....15
5.3.3 字段描述.....16
5.3.4 DMS_SER_CONFIG 串口高级选项定义.....17
6 设备配置设置.....21
6.1 功能描述.....21
6.2 DMSG_CONFIG_SET 消息.....21
7 设备复位.....22
7.1 功能描述.....22
7.2 DMSG_REBOOT 消息.....22
7.2.1 数据结构.....22
7.2.2 字段描述.....22

1 概述

1.1 目的

亚册 SDLC-ETH 协议转换器的管理编程接口，使用户应用程序能够、运行监控等功能，与产品无缝整合。

1.2 适用产品

- SDLC-ETH-200
- SDLC-ETH-201
- SDLC-ETH-400
- SDLC-ETH-401
- SDLC-CPCI/PXI
- SDLC-PCIe
- SDLC-SCM

本文统称 **SDLC-ETH**。

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 **SDLC-ETH 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，SDLC-ETH 管理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 DMS 接口

2.1 DMS

一般情况下，用户在 PC 机上安装 yacer-DMS 软件，构成 DMS 系统。

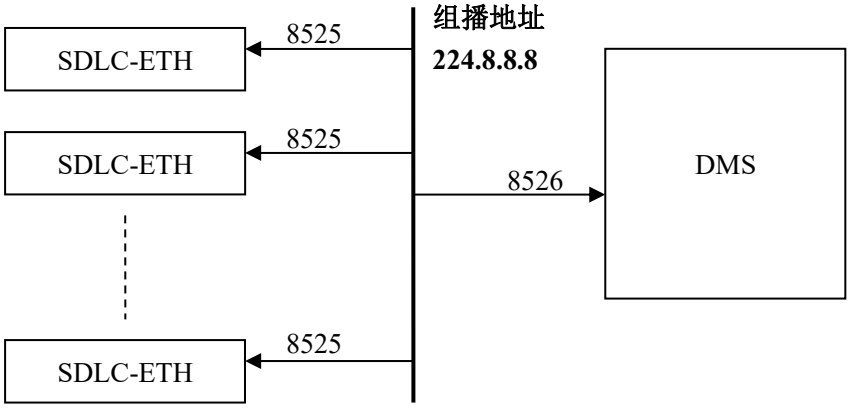
另外，用户也可以利用开发管理接口，自己编程实现 DMS 功能。

上述两种情况构建的 SDLC-ETH 管理系统，本文统称为 DMS。

2.2 通信方式

DMS 通过以太网和多台 SDLC-ETH 设备互联，基于 UDP 组播交互管理消息。

组播地址为 **224.8.8.8**，DMS 侧的 UDP 接收端口为 **8526**，SDLC-ETH 设备侧的 UDP 接收端口为 **8525**。



2.3 DMS 消息

DMS 与 SDLC-ETH 设备之间，通过 UDP 报文携带管理消息进行交互，传递配置信息、工作状态，这些消息统称为 DMS 消息。

DMS 消息由 message header（消息头）和 message body（消息体）组成：

消息头 Message header	消息体 Message body

2.4 DMSG_HEADER：DMS 消息头

2.4.1 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
}
```

2.4.2 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

2.4.3 消息头填写建议

DMS 发送时消息头填写建议如下：

字段	值	描述
dmsFlag	0x444D	
dmsVersion	0x20	
reserved	0x00	
sndDevType	0x10000000	发送方为 DMS 设备类型
sndDevSN	0x00000001 ~ 0xFFFFFFFE	用户在取值范围内任意指定，注意 ● 不能取 0 ● 不能取全 1 ● 如果同一网络存在多个 DMS，其 sndDevSN 不能够相同
rcvDevType	0x00007060	接收方为 SDLC-ETH-201/401
rcvDevSN	0xFFFFFFFF	● SDLC-ETH 的 SN 编号，通常由 DMSG_SEARCH_ACK 获得 ● 可设为全 1，则所有的 SDLC-ETH 均能够收到询问消息
msgType		消息类型
msgLen		填入消息字节数（含消息头）

2.4.4 设备类型编码

设备类型编码	描述
0xFFFFFFFF	所有类型
0x00007040	SDLC-ETH-200/400，SDLC-PCIe
0x00007060	SDLC-ETH-201/401，SDLC-CPCI/PXI，SDLC-SCM
0x10000000	DMS

2.4.5 消息类型

消息名称	类型编码	描述
DMSG_SEARCH_ID	0x0010	设备搜索消息
DMSG_SEARCH_ACK_ID	0x0011	设备搜索响应
DMSG_REPORT_GET_ID	0x1201	设备报告查询
DMSG_REPORT_GET_ACK_ID	0x1202	设备报告查询响应
DMSG_CONFIG_GET_ID	0x1000	设备配置查询
DMSG_CONFIG_GET_ACK_ID	0x1001	设备配置查询响应
DMSG_CONFIG_SET_ID	0x1100	设备配置设置
DMSG_REBOOT_ID	0x5A5A	设备复位命令

宏定义如下：

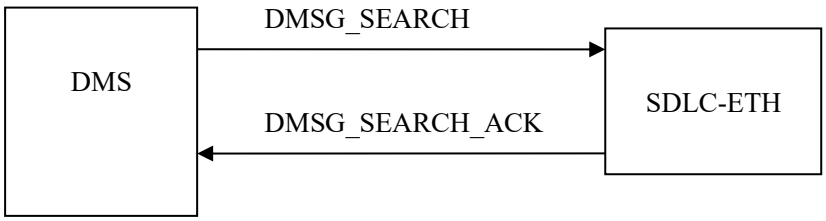
```
/* 设备搜索及反馈 */
#define DMSG_SEARCH_ID              0x0010
#define DMSG_SEARCH_ACK_ID          0x0011

/* 设备报告消息 */
#define DMSG_REPORT_GET_ID          0x1201
#define DMSG_REPORT_GET_ACK_ID      0x1202

* 设备配置消息 */
#define DMSG_CONFIG_GET_ID          0x1000      // 获取当前运行配置
#define DMSG_CONFIG_GET_ACK_ID      0x1001
#define DMSG_CONFIG_SET_ID          0x1100

/* 设备复位 */
#define DMSG_REBOOT_ID              0x5A5A
```

3 设备搜索及状态接口

3.1 功能描述

DMS 通过组播发送 DMSG_SEARCH 消息，同一网络中的 SDLC-ETH 设备收到后，返回 DMSG_SEARCH_ACK 消息。

通过该消息交互，DMS 服务器能够发现网络中在线的设备状态和基本信息，方便用户的配置和管理。



3.2 DMSG_SEARCH 消息

3.2.1 数据结构

结构定义如下：

```
struct DMSG_SEARCH
{
    struct   DMSG_HEADER    header;
    uint32_t      reserved;   // 保留
}
```

3.2.2 字段描述

字段定义如下：

字段	类型	描述
header	DMSG_HEADER	消息头，DMSG_HEADER
reserved	uint32_t	保留字段

3.3 DMSG_SEARCH_ACK 消息

3.3.1 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

    char reserved256[256];
};
```

3.3.2 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

3.3.3 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。

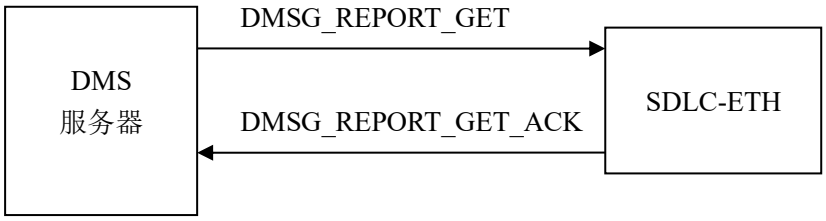
```
#define YC_SDLCER_NONE          0
#define YC_SDLCER_CLOCK         0x00000001UL   // 时钟故障
#define YC_SDLCER_RAM           0x00000002UL   // 内存故障
#define YC_SDLCER_XADC          0x00000004UL   // 温度传感器故障
#define YC_SDLCER_FLASH         0x00000008UL   // 闪存故障
#define YC_SDLCER_GPIO          0x00000010UL   // IO 端口故障
#define YC_SDLCER_PHY           0x00000020UL   // 以太网 PHY 故障
#define YC_SDLCER_UART          0x00000040UL   // 异步串口故障
#define YC_SDLCER_FPGA          0x00000080UL   // FPGA 故障
#define YC_SDLCER_UDPCLT        0x00000100UL   // UDP Client 异常
#define YC_SDLCER_UDPSVR        0x00000200UL   // UDP Server 异常
#define YC_SDLCER_PLL           0x00000400UL   // PLL 故障
#define YC_SDLCER_BAUDRATE      0x00000800UL   // 波特率异常
#define YC_SDLCER_TCPIP_MEM     0x00001000UL   // TCPIP 协议栈内存错误
```

```
#define YC_SDLCER_DFTSET        0x04000000UL  // 只能以默认配置运行提示
#define YC_SDLCER_DNA           0x08000000UL  // 设备 ID 校验失败
#define YC_SDLCER_FACTORY       0x10000000UL  // 设备出厂配置异常
#define YC_SDLCER_CONFIG        0x20000000UL  // 设备配置失败
#define YC_SDLCER_HWVER         0x40000000UL  // 硬件版本号异常
#define YC_SDLCER_SN            0x80000000UL  // 序列号异常
```

4 设备报告查询接口

4.1 功能描述

DMS 发送 DMSG_REPORT_GET 组播消息，网络中所有 SDLC-ETH 设备均能收到该询问消息。SDLC-ETH 检查消息中的 rcvDevSN 字段，如果同本设备编号一致，或者 rcvDevSN 为全 1 时，返回 DMSG_REPORT_GET_ACK 消息，该消息携带本设备工作状态报告。



4.2 DMSG_REPORT_GET 消息

4.2.1 数据结构

数据结构定义如下：

```
struct   DMSG_REPORT_GET
{
    DMSG_HEADER  header;
    uint32_t  statClearAfterReport;
}
```

4.2.2 字段描述

DMSG_REPORT_GET 消息字段按顺序定义为：

字段	类型	字节数	描述
header	DMSG_HEADER	24	消息头，DMSG_HEADER
statClearAfterReport	uint32_t	4	● 1：报告返回后，设备统计自动清零 ● 0：忽略

4.3 DMSG_REPORT_GET_ACK 消息

4.3.1 定义

```
#define YC_SDLC_SER_NUM            4   // 最大串口数量
#define YC_SDLC_UDPCLT_NUM        16   // UDP client 数量
#define YC_SDLC_UDPSVR_NUM        16   // UDP Server 数量
#define YC_SDLC_MULTICAST_NUM     16   // 加入的组播数量

/*************************************************************************
* 串口状态，位掩码方式
*/
#define YC_SDLC_SER_OK            0x00    // 串口正常
#define YC_SDLC_SER_BAUDRATE      0x01    // 串口波特率异常
#define YC_SDLC_SER_PLL           0x02    // 串口 PLL 异常
```

4.3.2 数据结构

数据结构定义：

```
struct DMSG_REPORT_GET_ACK
{
    DMSG_HEADER header;

    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_SDLC_SER_NUM];
    uint32_t ser_tx_overflow [YC_SDLC_SER_NUM];
    uint32_t ser_tx_toolong [YC_SDLC_SER_NUM];
    uint32_t ser_rx_pkt [YC_SDLC_SER_NUM];
    uint32_t ser_rx_crc_error [YC_SDLC_SER_NUM];
    uint32_t ser_rx_overflow [YC_SDLC_SER_NUM];
    uint32_t ser_rx_tooshort [YC_SDLC_SER_NUM];
    uint32_t ser_rx_toolong [YC_SDLC_SER_NUM];

    // UDP client
    uint32_t udp_tx_pkt [YC_SDLC_UDPCLT_NUM];
    uint32_t udp_tx_fail [YC_SDLC_UDPCLT_NUM];
```

```
        // UDP Server
        uint32_t udp_rx_pkt [YC_SDLC_UDPSVR_NUM];
        uint32_t udp_rx_fail [YC_SDLC_UDPSVR_NUM];

        uint32_t reserved4 [4];     //  保留字段

        //  串口真实波特率
        uint32_t    ser_realbd [YC_SDLC_SER_NUM];     // 0 表示波特率无效
        uint8_t    ser_status [YC_SDLC_SER_NUM];      //  串口状态

        uint8_t    reserved72 [72];    //  保留字段

        // APP_CPU1 信息
        uint32_t cpu1_runSeconds;     // cpu1 运行秒数
        uint32_t cpu1_status;           // cpu1 状态码，具体含义有用户确定
        uint32_t cpu1_udp_tx_cnt;
        uint32_t cpu1_udp_tx_overflow;
        uint32_t cpu1_udp_rx_cnt;
        uint32_t cpu1_ser_tx_cnt;
        uint32_t cpu1_ser_tx_overflow;
        uint32_t cpu1_ser_rx_cnt;
        char cpu1_status_string[128];// cpu1 状态字符串
};
```

4.3.3 字段描述

DMSG_REPORT_GET_ACK 消息字段按顺序定义为：

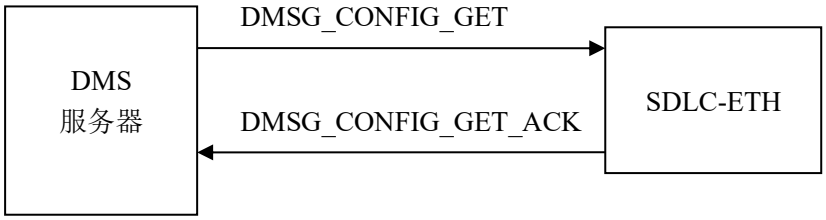
字段	类型	描述
header	DMSG_HEADER	消息头，DMSG_HEADER
runSeconds	uint32_t	设备加电后运行秒数
dms_tx_pkt	uint32_t	设备发送 DMS 消息计数
dms_tx_fail	uint32_t	设备发送 DMS 消息失败计数
dms_rx_pkt	uint32_t	设备接收 DMS 消息计数
dms_rx_invalid	uint32_t	设备接收无效 DMS 消息计数
ser_tx_pkt [ ]	uint32_t [ ]	串口发帧计数
ser_tx_overflow [ ]	uint32_t [ ]	串口发送溢出计数
ser_tx_toolong [ ]	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
ser_realbd [ ]	uint32_t [ ]	串口波特率
ser_status [ ]	uint8_t [ ]	串口状态
cpu1_runSeconds	uint32_t	APP_CPU 运行秒数
cpu1_status	uint32_t	APP_CPU 运行状态码，含义由用户定义
cpu1_udp_tx_cnt	uint32_t	APP_CPU 发送 UDP 报文计数
cpu1_udp_tx_overflow	uint32_t	APP_CPU 发送 UDP 报文溢出计数
cpu1_udp_rx_cnt	uint32_t	APP_CPU 接收 UDP 报文计数
cpu1_ser_tx_cnt	uint32_t	APP_CPU 串口发送计数
cpu1_ser_tx_overflow	uint32_t	APP_CPU 串口发送溢出计数
cpu1_ser_rx_cnt	uint32_t	APP_CPU 串口接收计数
cpu1_status	char [ ]	APP_CPU 状态描述字符串

5 设备配置查询接口

5.1 功能描述

DMS 发送 DMSG_CONFIG_GET 组播消息，网络中所有 SDLC-ETH 设备均能收到该询问消息。SDLC-ETH 检查消息中的 rcvDevSN 字段，如果同本设备编号一致，或者 rcvDevSN 为全 1 时，返回 DMSG_CONFIG_GET_ACK 消息，该消息携带本设备配置信息。



5.2 DMSG_CONFIG_GET 消息

5.2.1 数据结构

结构定义如下：

```
struct DMSG_CONIFG_GET
{
    struct   DMSG_HEADER    header;
    uint32_t      reserved;   // 保留
}
```

5.2.2 字段描述

字段定义如下：

字段	类型	描述
header	DMSG_HEADER	消息头，DMSG_HEADER
reserved	uint32_t	保留字段

5.3 DMSG_CONFIG_GET_ACK 消息

5.3.1 定义

```
#define YC_SDLC_SER_NUM            4   // 最大串口数量
#define YC_SDLC_UDPCLT_NUM        16   // UDP client 数量
#define YC_SDLC_UDPSVR_NUM        16   // UDP Server 数量
#define YC_SDLC_MULTICAST_NUM     16   // 加入的组播数量
```

5.3.2 数据结构

```
struct DMSG_CONFIG_GET_ACK
{
    DMSG_HEADER header;

    char deviceAlias[32];                    // 设备别名，用户定义名称

    // IP 接口配置
    uint32_t netif_ipaddr;
    uint32_t netif_netmask;
    uint32_t netif_gateway;

    // 串口配置
    uint8_t    ser_work_mode[YC_SDLC_SER_NUM];      // 工作模式
    uint8_t    ser_phy_mode[YC_SDLC_SER_NUM];       // 接口类型
    uint8_t    ser_terminal[YC_SDLC_SER_NUM];       // 终端电阻使能
    uint8_t    ser_reserved[YC_SDLC_SER_NUM];       // reserved
    uint32_t ser_baudrate[YC_SDLC_SER_NUM];         // 波特率
    DMS_SER_CONFIG ser_config[YC_SDLC_SER_NUM]; // 串口高级选项

    // UDP client
    uint8_t    udpclt_enable[YC_SDLC_UDPCLT_NUM];
    uint8_t    udpclt_src_lid[YC_SDLC_UDPCLT_NUM];      // 源串口逻辑编号 0~3
    uint16_t udpclt_tx_port[YC_SDLC_UDPCLT_NUM];       // 目的端口
    uint32_t udpclt_tx_ip[YC_SDLC_UDPCLT_NUM];         // 目的 IP 地址

    // UDP server
    uint8_t    udpsvr_enable[YC_SDLC_UDPSVR_NUM];
    uint8_t    udpsvs_dest_lid[YC_SDLC_UDPSVR_NUM];     // 目的串口逻辑编号 0~3
    uint16_t udpsvr_rx_port[YC_SDLC_UDPSVR_NUM];       // 接收 UDP 端口

    // 加入的 multicast address
    uint32_t multicast_ips[YC_SDLC_MULTICAST_NUM];

    // 串口转串口
    uint8_t    s2s_dest_mask[YC_SDLC_SER_NUM];

    // 串口发送缓冲区数量
    uint8_t    ser_txbuf_num;

    // 以太网桥接，收发串口分别指定
    uint8_t    eth_bridge_enable; // 桥接功能使能
```

```
    uint8_t    eth_bridge_txser;    // 用于以太网桥接的发送串口
    uint8_t    eth_bridge_rxser;    // 用于以太网桥接的接收串口

    uint32_t crc;
};
```

5.3.3 字段描述

DMSG_CONFIG_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
deviceAlias	char []	设备别名，用户定义的设备名称
netif_ipaddr	uint32_t	设备 IP 地址
netif_netmask	uint32_t	设备子网掩码
netif_gateway	uint32_t	默认网关
ser_work_mode	uint8_t []	串口工作模式： ● 0x00：异步 UART ● 0x01：同步 HDLC，NRZ 编码 ● 0x02：同步 HDLC，NRZI 编码 ● 0x04：异步 HDLC ● 0x05：同步 HDLC，DBPL 编码 ● 0x06：同步 HDLC，差分曼彻斯特编码 ● 0x07：同步 HDLC，曼彻斯特编码 ● 0x80：基于时钟的同步 bit 流采样 ● 0x81：基于多倍波特率的数据采样
ser_phy_mode	uint8_t []	串口接口类型： ● 6：RS-232 ● 5：RS-422 全双工 ● 10：RS-485 半双工
ser_terminal	uint8_t []	终端电阻匹配使能，1 表示使能
ser_reserved		保留
ser_baudrate	uint32_t []	串口波特率，单位 bps
ser_config	DMS_SER_CONFIG []	串口高级选项
udpclt_enable	uint8_t []	UDP Client（串口转 UDP）使能，1 表示使能
udpclt_src_lid	uint8_t []	串口转 UDP 源串口逻辑编号，0～3 表示 S1~S4
udpclt_tx_port	uint16_t []	串口转 UDP 目的端口
udpclt_tx_ip	uint32_t []	串口转 UDP 目的 IP 地址
udpsvr_enable	uint8_t []	UDP Server（UDP 转串口）使能，1 表示使能
udpsvs_dest_lid	uint8_t []	UDP 转串口目的串口编号，0～3 表示 S1~S4
udpsvr_rx_port	uint16_t []	UDP 转串口，接收 UDP 端口
multicast_ips	uint32_t []	UDP 组播接收地址
s2s_dest_mask	uint8_t []	串口转串口位掩码

ser_txbuf_num	uint8_t	串口发送缓冲区数量，默认 64
eth_bridge_enable	uint8_t	以太网桥接功能，1 表示使能
eth_bridge_txser	uint8_t	用于以太网桥接的发送串口编号
eth_bridge_rxser	uint8_t	用于以太网桥接的接收串口编号
crc	uint32_t	保留，未用

5.3.4 DMS_SER_CONFIG 串口高级选项定义

DMS_SER_CONFIG 串口高级选项，其数据结构定义随 ser_work_mode（工作模式）变化。

5.3.4.1 异步 UART 工作模式

常规异步串口的工作模式结构定义如下：

```
struct DMS_SER_UART
{
    uint16_t pkt_ms;        // 串口转网络打包时间，单位为 ms
    uint16_t pkt_len;       // 分包长度，单位为字节
    uint8_t data_bits;      // 数据位
    uint8_t parity;         // 校验
    uint8_t stop_bits;      // 停止位
};
```

字段	类型	描述
pkt_ms	uint16_t	串口转 UDP 时的分包时间，单位 ms
pkt_len	uint16_t	串口转 UDP 时的分包长度，单位为字节
data_bits	uint8_t	数据位定义： ● 3：8 bits ● 2：7 bits ● 1：6 bit ● 0：5 bits
parity	uint8_t	校验位定义： ● 4：Mark ● 3：Space ● 2：Even ● 1：Odd ● 0：None
stop_bits	uint8_t	● 停止位定义： ● 2：1.5 ● 1：2 ● 0：1

5.3.4.2 HDLC-NRZ 工作模式

采用 NRZ 编码格式的同步 HDLC 工作模式，定义如下：

```
struct DMS_SER_HDLC
{
    uint8_t   clock_mode;     // 时钟模式：NORMAL | SLAVE | MASTER
    uint8_t   tx_dchg;        // 发送触发：时钟上升沿 | 下降沿  产生新数据
    uint8_t   rx_dspl;        // 接收触发：时钟上升沿 | 下降沿  采样数据
    uint8_t   preamble_flag;  // 前导标志，位于 BOF 之前
    uint8_t   crc;
    uint8_t   fcs_forward;    // FCS 转发定义
    uint8_t   idle_flag;      // 空闲标志
    uint8_t   preamble_num;   // 前导标志个数
    uint8_t   head_data[4];   // 帧头数据内容
    uint8_t   head_size;      // 帧头长度
};
```

字段	类型	描述
clock_mode	uint8_t	时钟模式： ● 0：常规模式，发送时钟为自身产生的 TxC，接收用来自对端的 RxC ● 1：从模式（外时钟），发送、接收均用来自对端的 RxC，同时 TxC 环回 RxC 时钟，本端波特率设置无效 ● 2：主时钟模式，发送、接收均用自身产生的 TxC，忽略来自对端的 RxC 时钟
tx_dchg	uint8_t	发送触发： ● 0：时钟上升沿产生新数据 ● 1：时钟下降沿产生新数据
rx_dspl	uint8_t	接收触发： ● 0：时钟上升沿采样数据 ● 1：时钟下降沿采样数据
preamble_flag	uint8_t	前导标志，HDLC 协议可设置为 0x7E 或 0xFF
crc	uint8_t	CRC 功能： ● 0：禁用 CRC ● 1：16 位 CRC ISO 模式，HDLC 协议默认选项 ● 2：16 位 CRC IBM 模式，SDLC 协议选择 ● 3：32 位 CRC
fcs_forward	uint8_t	接收帧 FCS 字段转发模式： ● 0：抛弃 FCS 字段，不予转发 ● 1：数据转发时携带 FCS 字段
idle_flag	uint8_t	空闲标志：

		● 0：空闲填充 0xFF ● 1：空闲填充 0x7E
preamble_num	uint8_t	前导数量
head_data	uint8_t []	帧头内容，最多 4 字节
head_size	uint8_t	帧头长度： ● 0：无帧头附加 ● 1~4：非 HDLC->HDLC 增加帧头数据，HDLC->非 HDLC 删除帧头数据

5.3.4.3 HDLC-NRZI 工作模式

采用 NRZI 编码格式的同步 HDLC 工作模式，定义如下：

```
struct DMS_SER_HDLC_NRZI
{
    uint8_t   preamble_flag;
    uint8_t   crc;
    uint8_t   fcs_forward;
    uint8_t   reserved;
    uint8_t   preamble_num;
};
```

其字段含义同 HDLC-NRZ 工作模式。

5.3.4.4 HDLC-DBPL 工作模式

采用 DBPL（Differential Bi-Phase-Level）编码格式的同步 HDLC 工作模式，定义如下：

```
struct DMS_SER_HDLC_DBPL
{
    uint8_t   preamble_flag;
    uint8_t   crc;
    uint8_t   fcs_forward;
    uint8_t   reserved;
    uint8_t   preamble_num;
};
```

其字段含义同 HDLC-NRZ 工作模式。

5.3.4.5 HDLC-MAN 工作模式

采用曼彻斯特编码格式的同步 HDLC 工作模式，定义如下：

```
struct DMS_SER_HDLC_MAN
{
    uint8_t   preamble_flag;
    uint8_t   rising_edge;
    uint8_t   crc;
    uint8_t   fcs_forward;
    uint8_t   reserved;
    uint8_t   preamble_num;
}
```

字段定义：

字段	类型	描述
rising_edge	uint8_t	低->高跳变（上升沿）表示： ● 0：bit 0 ● 1：bit 1
其他		同 HDLC-NRZ 工作模式

5.3.4.6 HDLC-DiffMAN 工作模式

采用差分曼彻斯特编码格式的同步 HDLC 工作模式，定义如下：

```
struct DMS_SER_HDLC_DIFFMAN
{
    uint8_t   preamble_flag;
    uint8_t   crc;
    uint8_t   fcs_forward;
    uint8_t   reserved;
    uint8_t   preamble_num;
}
```

其字段含义同 HDLC-NRZ 工作模式。

5.3.4.7 同步 Bit 流模式

基于时钟的同步 bit 流采样模式，定义如下：

```
struct DMS_SER_BITS
{
    uint8_t   clock_mode;
    uint8_t   tx_dchg;
    uint8_t   rx_dspl;
    uint8_t   reserved1;
    uint8_t   msb_first;
    uint8_t   reserved2;
    uint16_t pkt_len;          // 分包长度，单位为字节
}
```

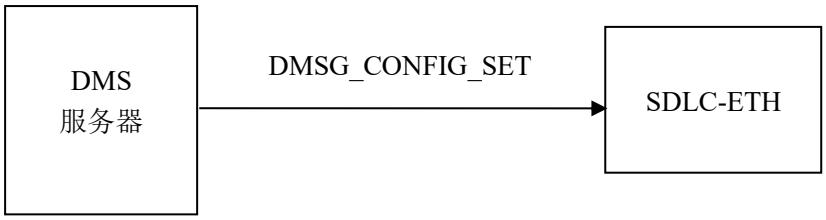
字段定义如下：

字段	类型	描述
msb_first	uint8_t	接收 bit 装配到字节中的顺序： ● 0：先收到的 bit 放在字节的低位 ● 1：先收到的 bit 放在字节的高位
pkt_len	uint16_t	接收到 pkt_len 字节长度的数据后，转发给网络
其他		同 HDLC-NRZ 工作模式

6 设备配置设置

6.1 功能描述

DMS 发送 DMSG_CONFIG_ST 组播消息，网络中所有 SDLC-ETH 设备均能收到该设置消息。SDLC-ETH 检查消息中的 rcvDevSN 字段，如果同本设备编号一致，或者 rcvDevSN 为全 1 时，写入新的配置至 FLASH，然后自动复位以使配置生效。



6.2 DMSG_CONFIG_SET 消息

定义同 DMS_CONFIG_GET_ACK 消息。

7 设备复位

7.1 功能描述

DMS 发送 DMSG_REBOOT 组播消息，网络中所有 SDLC-ETH 设备均能收到该设置消息。SDLC-ETH 检查消息中的 rcvDevSN 字段，如果同本设备编号一致，或者 rcvDevSN 为全 1 时，设备自动复位。

7.2 DMSG_REBOOT 消息

7.2.1 数据结构

结构定义如下：

```
struct DMSG_REBOOT
{
    struct   DMSG_HEADER    header;
    uint32_t      reserved;   // 保留
}
```

7.2.2 字段描述

字段定义如下：

字段	类型	描述
header	DMSG_HEADER	消息头，DMSG_HEADER
reserved	uint32_t	保留字段