CAN发送帧数据格式																							
帧头		帧长	命令	发送次数				时间间隔				ID类型	CAN ID				帧类型	idAcc	dataAcc	len	data[len]		CRC
Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Data16	Data17	Data18	Data19	Data20	Data21~Data28		Data29
16bit		8bit	8bit	32bit				16bit				8bit	32bit				8bit			8bit	8bit*8		8bit
55	AA	1e	01	01	00	00	00	0a	00	00	00	00	00	00	00	00	00	00	00	08	data[8]		crc

注：其余格式为 串口转发数据

命令	
0x01	转发CAN数据帧
0x02	PC 与设备握手，设备反馈OK
0x03	非反馈CAN转发，不反馈发送状态

ID类型：00 标准帧； 01扩展帧；
帧类型：00 数据帧；01远程帧

eg: 55 aa 1e 01 01 00 00 00 0a 00 00 00 00 00 00 00 00 00 08 00 00 12 23 34 45 56 67 78 89 88

串口波特率设置命令										
帧头		baudrate				databit	stopbit	parity	帧尾	
Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9	Data10
16bit		32bit				8bit	8bit	8bit	16bit	
0x55	0xAA								0xAA	0x55

```
typedef struct {
    uint32_t  baudrate;    //波特率
    uint8_t   databit;     //数据位
    uint8_t   stopbit;     //格式停止位
    uint8_t   parity;      //校验
}Baudrate_set_data;
```

```
//数据长度设定位
    switch(setdata_t->databit)  //0: 8bit  1: 9bit
    {
        case 0u:  stcInitCfg.enDataLength = UsartDataBits8; break;
        case 1u:  stcInitCfg.enDataLength = UsartDataBits9; break;
        default: stcInitCfg.enDataLength = UsartDataBits8; break;
    }
    //奇偶校验设置
    switch(setdata_t->parity) //
    {
        case 0u:  stcInitCfg.enParity = UsartParityNone; break;
        case 1u:  stcInitCfg.enParity = UsartParityEven; break;
        case 2u:  stcInitCfg.enParity = UsartParityOdd;  break;
        default:  stcInitCfg.enParity = UsartParityNone; break;
    }

    //奇偶校验设置
    switch(setdata_t->stopbit) //
    {
        case 0u:  stcInitCfg.enStopBit = UsartOneStopBit; break;
        case 1u:  stcInitCfg.enStopBit = UsartTwoStopBit; break;
        default:  stcInitCfg.enStopBit = UsartTwoStopBit; break;
    }
```

设置CAN 波特率命令				
帧头		索引	帧尾	
Data0	Data1	Data2	Data3	Data4
16bit		8bit		
0x55	0x05		0xAA	0x55

```
索引     波特率       理想CIA    实际CIA
//index0----1000kbps    -- 75%      -- 75  %
//index1---- 800kbps    -- 80%      -- 80  %
//index2---- 666kbps    -- 80%      -- 83.3%
//index3---- 500kbps    -- 87.5%    -- 87.5%
//index4---- 400kbps    -- 87.5%    -- 85  %
//index5---- 250kbps    -- 87.5%    -- 87.5%
//index6---- 200kbps    -- 87.5%    -- 85  %
//index7---- 125kbps    -- 87.5%    -- 87.5%
//index8---- 100kbps    -- 87.5%    -- 87.5%
//index9----  80kbps    -- 87.5%    -- 865%
//index10---  50kbps    -- 87.5%    -- 87.5%
//index11---  40kbps    -- 87.5%    -- 85  %
//index12---  20kbps    -- 87.5%    -- 85  %
//index13---  10kbps    -- 87.5%    -- 85  %
//index14---   5kbps    -- 87.5%    -- 87.5%
```

PC与设备心跳					
帧头		帧尾			
Data0	Data1	Data3			Data4
16bit					
0x55	0x04	0xAA			0x55

注： CAN返回数据 CMD为00 其余can数据无效

CAN停止发送（连续发送中）					
帧头		帧尾			
Data0	Data1	Data3			Data4
16bit					
0x55	0x03	0xAA			0x55

注：停止现有连续发送

CAN接收数据帧格式															
帧头	命令	格式	CANID				数据								帧尾
Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15
AA															55

CAN 命令 //00 心跳 0x01 接收失败 0x11 接收成功 0x02 发送失败 0x12 发送成功 0x03 波特率设置失败 0x13 波特率设置成功
格式 包含数据长度 数据帧类型 详见如下结构体

```
CAN 返回数据格式
typedef struct                    //can发送功能相关结构体  16bytes
{
    uint8_t     freamHeader;     //发送标志位 0xAA
    uint8_t     CMD;              //CAN 命令 //00 心跳 0x01 接收失败 0x11 接收成功  0x02 发送失败 0x12 发送成功
    uint8_t     canDataLen:6;   //数据长度
    uint8_t     canIde:1;         //ide:0,标准帧;1,扩展帧
    uint8_t     canRtr:1;         //rtr:0,数据帧;1,远程帧
    uint32_t    CANID;            //can ID
    uint8_t     canData[8];       //Can 数据
    uint8_t     freamEnd;         //结尾      0x55
} CAN_Fream;
```

eg: AA 11 08 00 00 00 00 00 01 02 03 04 05 06 07 55