HPS-166-L

高性能飞行时间(ToF)传感器

产品特性

- 快速、精确测距
  - 最远测量距离可达 25m (白色目标)
  - 自适应输出数据速率高达 54Hz
  - 测量结果对目标的颜色和反射率不敏感
  - 电气和光学串扰补偿
  - -10℃~+55℃ 温度补偿
  - 内置环境光补偿功能使得传感器能在高红外背景光环境下工作
- 完全集成的微型模块，包含:
  - 850nm 红外 VCSEL 发射器
  - 发射器驱动电路
  - 优化设计的发射和接收光学镜头
  - 高性能嵌入式微处理器
  - 先进的嵌入式数据处理与滤波算法
  - 115200bps 3.3V UART 串行接口
  - 34(长) x 24(宽) x 22(高) mm, 7g
  - 通过最新的 CE, FCC, RoHS 认证

产品应用

- 无人机 (避障，软着陆)
- 机器人 & AGV 自动导航机器人 (障碍物检测)
- 工业定位和接近传感器
- 安全和监控
- 1D 动作识别

产品描述

HPS-166-L 是新一代高性能飞行时间(ToF)红外测距传感器，配有优化设计的发射和接收光学镜头，适用于高精度、长距离的测距场合。区别于传统的技术，它的测量精度不受测量目标颜色和反射率的影响。目标为白色时，HPS-166-L 测量距离可以达到 25 米。

HPS-166-L 集成 1 个大功率 850nm 红外 VCSEL 发射器和 1 个高灵敏度光电二极管接收器，加上内部集成的物理红外滤光片，使其测距离更远，抗环境光干扰能力更强。

内部集成先进的嵌入式数据处理和滤波算法，实现了非常稳定和实时的测量结果输出。

订购信息

HPS- 1 6 x C - L

- 海伯森产品系列标识
- **16x**: ToF测距/避障传感器系列
- **无**: 3.3V UART接口，无防水外壳
  U: 3.3V UART接口，带防水外壳
  C: 1Mbps CAN总线接口，带防水外壳
- **无**: 850nm LED (发光二极管)发射器
  L: 850nm VCSEL (垂直腔面发射激光)发射器

CE FC RoHS

AVOID EYE OR SKIN EXPOSURE TO DIRECT OR SCATTERED RADIATION

概述

1.1 技术规格

表 1. 技术规格

参数	值	单位
尺寸	34（长）x 24（宽）x 22（高）*	mm
重量	7 *1	g
供电	4 ~ 6	V
最大功耗	1.3	W
静态功耗	0.1	W
存储温度	-40 ~ 85	℃
工作温度	-10 ~ 55*2	℃
红外 VCSEL 发射波长	850	nm
发射角度	±1.8	°
最大测量距离	25 *3	m
最小测量距离	0.08	m
自适应输出数据频率	6 ~ 54	Hz
输出数据	距离，精度指示，信号强度，环境光强，温度	-
插座	0.5mm 间距，8-针，FPC 接口，上下双面接触	-
接口	TTL UART, 115200bps, 8 data bits, no parity, 1 stop bit	

注： *1 不带镜头盖。

*2 在连续工作模式下进行远距离测量时，HPS-166-L 需要一定的预热时间以稳定输出。在长时间连续工作模式下，需要保证电路板一侧的散热，否则传感器过热将导致测量背景噪声升高。

*3 测试基于 90%反射率的白色目标。

1.2 外形尺寸和针脚定义

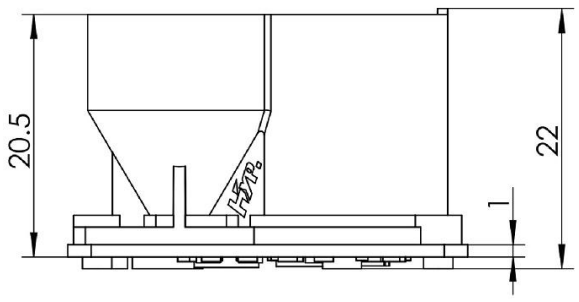

图 1. HPS-166-L 前视图

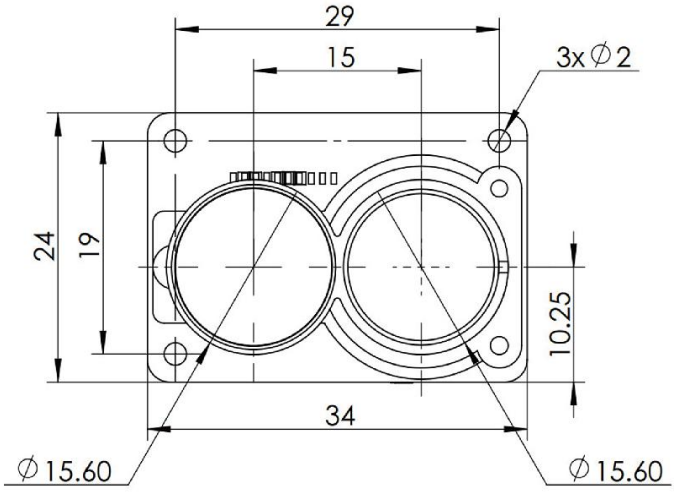

图 2. HPS-166-L 俯视图

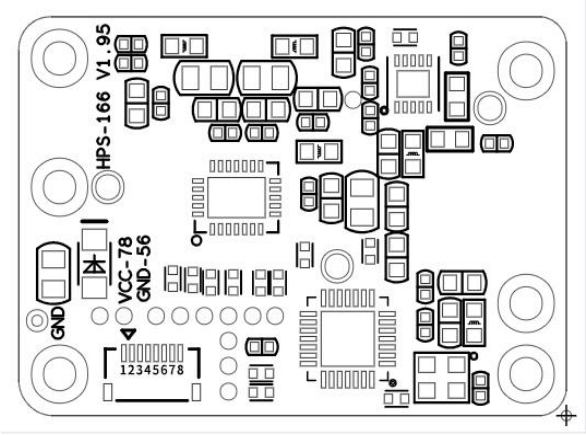

图 3. HPS-166-L 仰视图

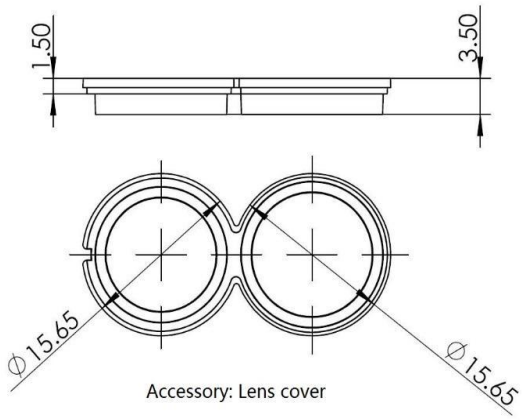

图 4. 镜头盖前视图和仰视图

表 2. HPS-166-L 针脚定义

针脚号	信号名称	信号种类	描述
1	VDD	Power	电源，连接到 DC +5V
2	VDD	Power	电源，连接到 DC +5V
3	GND	GND	电源地
4	GND	GND	电源地
5	INT	Digital output	中断信号输出（脉冲宽度：100 us）
6	RST	Digital input	复位信号，低电平有效
7	RXD	Digital input	UART TTL 输入端
8	TXD	Digital output	UART TTL 输出端

HPS-166-L的所有针脚均符合表3列出的IEC61000-4-2抗ESD测试标准：

表 3. ESD performances

Parameter	Conditions
Air Discharge	+/- 8kV
Direct Contact	+/- 4kV
Indirect Contact HCP	+/- 4kV
Indirect Contact VCP	+/- 4kV

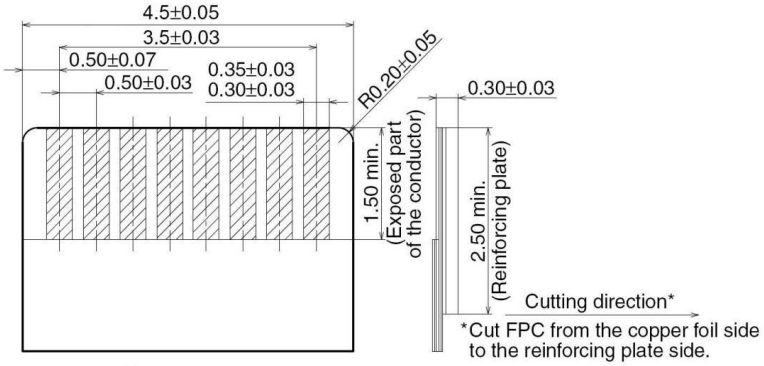

图 5. 推荐的 FPC/FFC 尺寸

1.3 极限参数

表 4. HPS-166-L 极限参数

Parameter	Min.	Typ.	Max.	Unit
VDD	-0.3	-	6.5	V
RXD, RST	-0.3	-	5.6	V

注：工作范围超过表4所列的极限参数可能会对传感器造成永久性的损坏或直接影响传感器的可靠性。

1.4 推荐参数

表 5. HPS-166-L 推荐参数

Parameter	Min.	Typ.	Max.	Unit
VDD	4	5	6	V
RXD, RST	2.8	3.3	3.6	V

1.5 连接方式

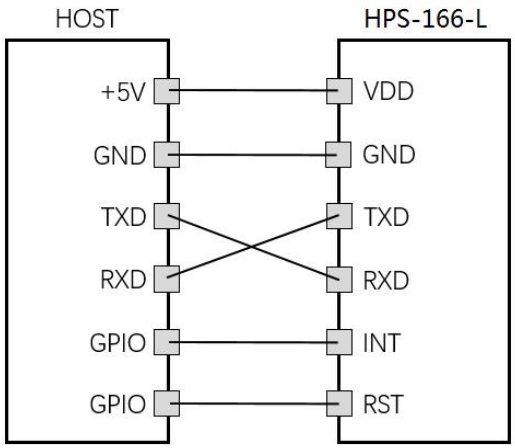

图6. HPS-166-L的连接方式

控制界面

2.1 TTL UART 串口

HPS-166-L 使用 TTL UART 串口与主机进行通讯，其逻辑电平与采用 3.3V 供电的系统相兼容。

表 6. UART 串口参数

波特率	115200bps
起始位	1bit
数据位	8bits
奇校验位	0bit
停止位	1bit

2.2 通讯协议

传感器上电后，系统会自动进行初始化，初始化成功后串口会输出字符串“Hypersen”。起始字节“0x0A”被用来指示每个命令和返回数据帧的开始。每个 HPS-166-L 传感器都有一个通用唯一识别码（UUID），该识别码可以通过向传感器发送相应的命令进行读取。

命令#1：获取传感器信息

表 7. 获取传感器信息命令

起始字节	命令字节	数据区域						CRC 高字节	CRC 低字节
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
0x0A	0x2E	0x00	0x00	0x00	0x00	0x00	0x00	0xFC	0x3C

返回数据：

表 8. 获取传感器信息命令的返回数据

字节号	名称	值	描述
0	Start byte	0x0A	返回数据的起始字节
1	Data length	0x18	数据区域的字节长度，该长度不包括第 0 和第 1 字节
2	ACK byte	0xB0	传感器返回的应答字节
3~18	UUID	……	传感器的通用唯一识别码
19	Year	……	产品的生产日期
20	Month	……	
21	Day	……	
22	Major version	……	传感器版本号
23	Minor version	……	
24	CRC MSB	……	当前数据帧的 CRC 计算值（第 2~23 字节）
25	CRC LSB	……	

以下是一帧返回的传感器信息数据：

0x0A 0x18 0xB0 0x52 0x13 0x29 0x8C 0xC7 0xE0 0xE5 0x11 0x8D 0x2B 0xB9 0x57 0x2C 0xF3 0xAD 0x25 0x10 0x0A 0x08 0x01 0x09 0x22 0xE9

解析：

0x0A: 起始字节

0x18: 数据长度（24 字节）

0xB0: 应答字节

0x52 0x13 0x29 0x8C 0xC7 0xE0 0xE5 0x11 0x8D 0x2B 0xB9 0x57 0x2C 0xF3 0xAD 0x25: 通用唯一识别码

0x10 0x0A 0x08: 16/10/08

0x01 0x09: Ver. 1.9

0x22 0xE9: CRC16-CCITT 的高字节和低字节

命令#2：连续测量

表 9. 连续测量命令

起始字节	命令字节	数据区域						CRC 高字节	CRC 低字节
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
0x0A	0x24	0x00	0x00	0x00	0x00	0x00	0x00	0x0F	0x72

命令#3：单次测量

表 10. 单次测量命令

起始字节	命令字节	数据区域						CRC 高字节	CRC 低字节
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
0x0A	0x22	0x00	0x00	0x00	0x00	0x00	0x00	0xAE	0x57

返回数据：

表 11. 测量结果的返回数据

字节号	名称	值	描述
0	Start byte	0x0A	返回数据的起始字节
1	Data length	0x0D	数据区域的字节长度，该长度不包括第 0 和第 1 字节
2~4	Reserved	……	保留字节
5	Distance MSB	……	测量结果，单位：毫米
6	Distance LSB	……	
7	Magnitude MSB	……	接收反射光信号强度
8	Magnitude LSB	……	
9	Magnitude Exp.	……	
10	Ambient ADC	……	相对环境红外光强度
11	Precision MSB	……	精度指示值，值越小代表测量精度越高
12	Precision LSB	……	
13	CRC MSB	……	当前数据帧的 CRC 计算值（第 2~12 字节）
14	CRC LSB	……	

注：当传感器超量程或反射信号强度过低时会输出 65.53 米的超量程指示。

以下是一帧返回的测量结果数据：

0x0A 0x0D 0x01 0x01 0x01 0x06 0xD9 0xFC 0x8C 0x02 0x01 0x00 0x01 0x9B 0x94

解析：

0x0A: 起始字节

0x0D: 数据长度（13 字节）

Distance = (0x06 * 256 + 0xD9) / 1000.0f = 1.753 (单位：米)

Magnitude = ((0xFC * 256 + 0x8C) << 0x02) / 10000.0f = 25.8608

Ambient ADC = 1

Precision = (0x00 * 256) + 0x01 = 1

0x9B 0x94: CRC16-CCITT 的高字节和低字节

命令#4：停止测量

表 12. 停止测量命令

起始字节	命令字节	数据区域						CRC 高字节	CRC 低字节
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
0x0A	0x30	0x01	0x00	0x00	0x00	0x00	0x00	0xBC	0x6F

返回数据：

表 13. 停止测量命令的返回数据

字节号	名称	值	描述
0	Start byte	0x0A	返回数据的起始字节
1	Data length	0x03	数据区域的字节长度，该长度不包括第 0 和第 1 字节
2	ACK	……	0x01: 成功; 0x00: 失败
3	CRC MSB	……	当前数据帧的 CRC 计算值（第 2 字节）
4	CRC LSB	……	

命令#5：设置测量距离偏差补偿值

表 14. 设置测量偏差补偿值命令

起始字节	命令字节	数据区域						CRC 高字节	CRC 低字节
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
0x0A	0x38	0x1A	偏差高字节	偏差低字节	0x00	0x00	0x00	CRC 高字节	CRC 低字节

测量偏差值 = 真实距离 – 传感器测量距离，单位：毫米

例子：

真实距离：200 毫米，传感器测量距离：215 毫米

测量偏差值 = 200 – 215 = -15 = 0xFFF1 (偏差值高字节 = 0xFF，偏差值低字节 = 0xF1)

注：由于传感器个体的性能差异，该命令可用来补偿小范围的测量偏差以实现更高的测量精度。调用该命令设定的偏差值会自动被保存到传感器的 Flash 存储器中并在每次传感器上电时自动加载。

返回数据：

表 15. 设置测量偏差补偿值命令的返回数据

字节号	名称	值	描述
0	Start byte	0x0A	返回数据的起始字节
1	Data length	0x03	数据区域的字节长度，该长度不包括第 0 和第 1 字节
2	ACK	……	0x01: 成功; 0x00: 失败
3	CRC MSB	……	当前数据帧的 CRC 计算值（第 2 字节）
4	CRC LSB	……	

命令#6：加载设置参数

表 16. 加载设置参数命令

起始字节	命令字节	数据区域						CRC 高字节	CRC 低字节
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
0x0A	0x3E	0x00: 用户设置参数 0xFF: 出厂设置参数	0x00	0x00	0x00	0x00	0x00	CRC 高字节	CRC 低字节

返回数据：

表 17. 加载设置参数命令的返回数据

字节号	名称	值	描述
0	Start byte	0x0A	返回数据的起始字节
1	Data length	0x03	数据区域的字节长度，该长度不包括第 0 和第 1 字节
2	ACK	……	0x01: 成功; 0x00: 失败
3	CRC MSB	……	当前数据帧的 CRC 计算值（第 2 字节）
4	CRC LSB	……	

命令#7：输出滤波器设置

表 18. 输出滤波器设置命令

起始字节	命令字节	数据区域						CRC 高字节	CRC 低字节
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
0x0A	0x3D	0xAA	滤波器高字节	滤波器低字节	0x00	0x00	0x00	CRC 高字节	CRC 低字节

输出滤波器默认值：0x0000

例子：

将输出数据稳定性降低 50 个单位 -> 输出滤波器值 = -50 = 0xFFCE (滤波器高字节 = 0xFF，滤波器低字节 = 0xCE)

将输出数据稳定性提高 50 个单位 -> 输出滤波器值 = 50 = 0x0032 (滤波器高字节 = 0x00，滤波器低字节 = 0x32)

返回数据：

表 19. 输出滤波器设置命令的返回数据

字节号	名称	值	描述
0	Start byte	0x0A	返回数据的起始字节
1	Data length	0x03	数据区域的字节长度，该长度不包括第 0 和第 1 字节
2	ACK	……	0x01：成功; 0x00：失败
3	CRC MSB	……	当前数据帧的 CRC 计算值（第 2 字节）
4	CRC LSB	……	

命令#8：获取模拟前端（AFE）的温度

表 20. 获取模拟前端（AFE）的温度命令

起始字节	命令字节	数据区域						CRC 高字节	CRC 低字节
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
0x0A	0x3F	0x01	0x00	0x00	0x00	0x00	0x00	0x36	0x86

返回数据：

表 21. 获取模拟前端（AFE）的温度命令的返回数据

字节号	名称	值	描述
0	Start byte	0x0A	返回数据的起始字节
1	Data length	0x04	数据区域的字节长度，该长度不包括第 0 和第 1 字节
2	AFE Temperature MSB	.....	AFE 温度 = (MSB*256 + LSB) / 100，（单位：华氏度）
3	AFE Temperature LSB	.....	
4	CRC MSB	……	当前数据帧的 CRC 计算值（第 2~3 字节）
5	CRC LSB	……	

包装信息

表 22. 包装规格

型号	HPS-166-L
传感器尺寸	34（长）x 24（宽）x 23.5（高）mm （包含镜盖高度）
重量	7.9 克 / 个 （包含镜盖重量）
托盘	50 个（10*5）每盘
外箱	4 盘 / 箱（200 个）

修订历史记录

表 23. 规格书修订历史记录

Date	Revision	Description
2016/12/23	1.0	初始版本。
2017/01/01	1.1	加入推荐的 FPC/FFC 尺寸描述。
2017/01/10	1.2	修改数据输出频率为：6~54 Hz。
2017/02/05	1.3	修改传感器针脚定义；加入了命令#5 ~ #7。
2017/02/20	1.4	将传感器图片更换为 V1.93 版硬件图片；修改了传感器针脚定义。
2017/02/24	1.5	加入命令#8；修正了 1.4 版本中传感器针脚顺序。
2017/03/07	1.6	在命令和返回数据描述中加入了 CRC 计算字节说明。
2017/03/31	1.7	修改了命令#8 及返回数据桢的格式。
2017/11/29	1.8	加入了超量程 65.53 米输出描述。
2018/01/16	1.9	加入了产品订购信息。
2021/04/19	2.0	修改了量程信息
2022/08/19	2.1	将传感器图片更换为 V1.95 版硬件图片
2023/06/08	2.2	图 3，V1.95 版硬件图片增加针脚号

附录

CRC16-CCITT 的 C 语言实现

实现 1:

```
################################################################
#include<stdio.h>
/**
Flash Space: Small
Calculation Speed: Slow
*/
/*Function Name:          crc_cal_by_bit          //按位计算CRC
  Function Parameters:    unsigned char* ptr      //数据缓冲区指针
                          unsigned char len       //数据长度
  Return Value:           unsigned int
  Polynomial:             CRC-CCITT 0x1021
*/
unsigned int crc_cal_by_bit(unsigned char* ptr, unsigned char len)
{
    #define CRC_CCITT  0x1021
    unsigned int crc = 0xffff;

    while(len-- != 0)
    {
        for(unsigned char i = 0x80; i != 0; i /= 2)
        {
            crc *= 2;
            if((crc&0x10000) !=0)
                crc ^= 0x11021;
            if((*ptr&i) != 0)
                crc ^= CRC_CCITT;
        }
        ptr++;
    }
    return crc;
}
################################################################
```

实现 2:

```
#################################################################
#include<stdio.h>
/**
Flash Space: Medium
Calculation Speed: Medium
*/
/* Function Name:         crc_cal_by_halfbyte    //按半字节计算CRC
   Function Parameters:   unsigned char* ptr     //数据缓冲区指针
                          unsigned char len      //数据长度
   Return Value:          unsigned int
   Polynomial:            CRC-CCITT  0x1021
*/
unsigned int crc_cal_by_halfbyte(unsigned char* ptr, unsigned char len)
{
    unsigned short crc = 0xffff;

    while(len-- != 0)
    {
        unsigned char high = (unsigned char)(crc/4096);
        crc <<= 4;
        crc ^= crc_ta_4[high^(*ptr/16)];
        high = (unsigned char)(crc/4096);
        crc <<= 4;
        crc ^= crc_ta_4[high^(*ptr&0x0f)];
        ptr++;
    }
    return crc;
}

unsigned int crc_ta_4[16]={ /* CRC半字节表 */
   0x0000,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,
   0x8108,0x9129,0xa14a,0xb16b,0xc18c,0xd1ad,0xe1ce,0xf1ef,
};
#################################################################
```

实现 3:

```
#####################################################################
#include<stdio.h>
/**
Flash Space: Large
Calculation Speed: Fast
*/
/* Function Name:          crc_cal_by_byte        //按字节计算CRC
   Function Parameters:    unsigned char* ptr     //数据缓冲区指针
                           unsigned char len      //数据长度
   Return Value:           unsigned int
   Polynomial:             CRC-CCITT 0x1021
*/
unsigned int crc_ta_8[256]={ /* CRC字节表 */
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
    0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
    0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
    0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
    0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
    0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
    0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
    0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
    0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
    0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
    0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
    0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
    0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
    0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
    0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
    0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
    0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
    0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
    0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
    0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
    0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
    0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
```

```
    0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
};

unsigned int crc_cal_by_byte(unsigned char* ptr, unsigned char len)
{
    unsigned short crc = 0xffff;

    while(len-- != 0)
    {
        unsigned int high = (unsigned int)(crc/256);
        crc <<= 8;
        crc ^= crc_ta_8[high^*ptr];
        ptr++;
    }

    return crc;
}
```

##

测试代码:

##

```
void main()
{
    unsigned char sample_data[] = {0x01, 0x01, 0x01, 0x06, 0xd9, 0xfc, 0x8c, 0x02, 0x01, 0x00,
0x01};//Result should be: 0x9b94
    unsigned char data1[] = {0x63};//Result should be: 0xbd35
    unsigned char data2[] = {0x8c};//Result should be: 0xb1f4
    unsigned char data3[] = {0x7d};//Result should be: 0x4eca
    unsigned char data4[] = {0xaa, 0xbb,0xcc};//Result should be: 0x6cf6
    unsigned char data5[] = {0x00,0x00,0xaa, 0xbb, 0xcc};//Result should be: 0xb166
    unsigned short r1 = 0, r2=0, r3=0, r4=0, r5=0, r_sample_data;

    //Implementation 1
    r1 = crc_cal_by_byte(data1, 1);
    r2 = crc_cal_by_byte(data2, 1);
    r3 = crc_cal_by_byte(data3, 1);
    r4 = crc_cal_by_byte(data4, 3);
    r5 = crc_cal_by_byte(data5, 5);
    r_sample_data = crc_cal_by_byte(sample_data, 11);
    printf("Implementation_1: r1= %x, r2=%x, r3=%x, r4=%x, r5=%x, r_sample_data=%x\n", r1, r2, r3,
r4, r5,r_sample_data);
    r1=r2=r3=r4=r5=0;
```

```
    //Implementation 2
    r1 = crc_cal_by_bit(data1, 1);
    r2 = crc_cal_by_bit(data2, 1);
    r3 = crc_cal_by_bit(data3, 1);
    r4 = crc_cal_by_bit(data4, 3);
    r5 = crc_cal_by_bit(data5, 5);
    r_sample_data = crc_cal_by_bit(sample_data, 11);
    printf("Implementation_2: r1= %x, r2=%x, r3=%x, r4=%x, r5=%x, r_sample_data=%x\n", r1, r2, r3,
r4, r5,r_sample_data);
    r1=r2=r3=r4=r5=0;

    //Implementation 3
    r1 = crc_cal_by_halfbyte(data1, 1);
    r2 = crc_cal_by_halfbyte(data2, 1);
    r3 = crc_cal_by_halfbyte(data3, 1);
    r4 = crc_cal_by_halfbyte(data4, 3);
    r5 = crc_cal_by_halfbyte(data5, 5);
    r_sample_data = crc_cal_by_halfbyte(sample_data, 11);
    printf("Implementation_3: r1= %x, r2=%x, r3=%x, r4=%x, r5=%x, r_sample_data=%x\n", r1, r2, r3,
r4, r5,r_sample_data);
    r1=r2=r3=r4=r5=0;
}
############################################################
```

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Hypersen Technologies Co., Ltd. reserve the right to make changes, corrections, enhancements, modifications, and improvements to Hypersen products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on Hypersen products before placing orders. Hypersen products are sold pursuant to Hypersen's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of Hypersen products and Hypersen assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by Hypersen herein.

Resale of Hypersen products with provisions different from the information set forth herein shall void any warranty granted by Hypersen for such product.

Hypersen and the Hypersen logo are trademarks of Hypersen. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.