Hypersen
HYPERSEN TECHNOLOGIES CO., LTD.

RS485 接口

# RS485 接口命令手册

## 控制界面

### 1、RS-485 总线通讯协议

RS485 默认通讯波特率为 115200bps，用户可通过发送命令修改波特率，RS485 接口支持 9600bps, 115200bps, 128000bps, 256000bps, 460800bps, 600000bps, 750000bps, 921600bps 和 1500000bps 九种波特率。

表 1. RS-485 参数

| | |
|---|---|
| 默认波特率 | 115200bps |
| 数据位 | 8bits |
| 奇偶校验 | 偶校验 |
| 停止位 | 1bit |
| CRC 校验 | CRC16-CCITT |

#### 1.1 数据帧格式

表 2. 传感器数据帧格式

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | 内容 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6~N | ByteN+1 | ByteN+2 | ByteN+3 | ByteN+4 |
| 0XF6 | 0X6F | 1 字节 | 0x00 (默认) | 0x00 | 1 字节 | 0~255 字节 | 1 字节 | 1 字节 | 0x6F | 0xF6 |

#### 1.2 命令列表

表 3. 传感器命令列表

| 命令 | 命令字节 | 描述 |
|---|---|---|
| 读取设备 ID | 0x01 | 读取设备的 ID 号: 0x46FE |
| 连续测量 | 0x02 | 连续输出测量数据 |
| 停止连续测量 | 0x03 | 停止连续测量 |
| 单次测量 | 0x04 | 单次输出测量数据 |
| 设置测量频率 | 0x05 | 传感器支持 0~4 五个测量频率档位，用户可根据应用进行选择 |
| 设置波特率 | 0x06 | 可设置 9600bps, 115200bps, 128000bps, 256000bps, 460800bps, 614400bps 和 921600bps 七种波特率 |
| 恢复用户设置 | 0x07 | 恢复 Flash 中保存的用户设置，包括测量频率、波特率、设备地址 |
| 恢复出厂设置 | 0x08 | 恢复出厂设置 |
| 保存用户设置 | 0x09 | 将用户设置保存到 Flash 中 |
| 获取版本信息 | 0x0A | 获取传感器的版本信息 |
| 零点校正 | 0x0B | 执行传感器零点校正 |
| 复位设备地址 | 0x0D | 将传感器的设备地址恢复到默认值（0x00） |

| | | |
|---|---|---|
| 设置设备地址 | 0x0E | 设置传感器的设备地址 |

**命令#1 获取传感器的设备 ID 号**

该命令可获取传感器的设备 ID 号（0x46FE）。

表 4. 获取传感器的设备 ID 号命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x01 | 0xBD | 0xDC | 0x6F | 0xF6 |

返回数据：

表 5. 获取传感器的设备 ID 号命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x05 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x01 | 命令字节 |
| 6 | 0xFE | 设备 ID 号低字节 |
| 7 | 0x46 | 设备 ID 号高字节 |
| 8 | 0xF0 | CRC16-CCITT 校验低字节 |
| 9 | 0x3E | CRC16-CCITT 校验高字节 |
| 10 | 0x6F | 帧尾第一字节 |
| 11 | 0xF6 | 帧尾第二字节 |

**命令#2 单次测量**

此命令将启动传感器进行一次单次测量。

表 6. 单次测量命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x04 | 0x18 | 0x8C | 0x6F | 0xF6 |

**命令#3 连续测量**

此命令将启动传感器进行连续测量。

表 7. 连续测量命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x02 | 0xDE | 0xEC | 0x6F | 0xF6 |

单次/连续测量命令返回数据：

表 8. 单次/续测量命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |

| | | |
|---|---|---|
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x1B | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 异常数据指示，0 表示数据正常、0xFF 表示数据出现异常 |
| 5 | 0x04/0x02 | 命令字节 |
| 6~9 | …… | Fx 的低字节~高字节的整型补码，该数值/1000=Fx （单位：N） |
| 10~13 | …… | Fy 的低字节~高字节的整型补码，该数值/1000=Fy （单位：N） |
| 14~17 | …… | Fz 的低字节~高字节的整型补码，该数值/1000=Fz （单位：N） |
| 18~21 | …… | Mx 的低字节~高字节的整型补码，该数值/1000=Mx （单位：Nm） |
| 22~25 | …… | My 的低字节~高字节的整型补码，该数值/1000=My （单位：Nm） |
| 26~29 | …… | Mz 的低字节~高字节的整型补码，该数值/1000=Mz （单位：Nm） |
| 30 | …… | CRC16-CCITT 校验低字节 |
| 31 | …… | CRC16-CCITT 校验高字节 |
| 32 | 0x6F | 帧尾第一字节 |
| 33 | 0xF6 | 帧尾第二字节 |

**数据解析例：**

以下为连续测量模式下的一帧测量数据:

F6 6F 1B 00 00 02 16 FF FF FF 01 FA FF FF EF 02 00 00 06 00 00 00 0A 00 00 00 0F 00 00 00 6F 58 6F F6

**解析：**

0xF6: 帧头

0x6F: 帧头

0x1B: 有效数据长度

0x00: 设备地址

0x00: 数据正常

0x02: 连续测量命令字节

0x16 0xFF 0xFF 0xFF: Fx 数据，0xFFFFFF16 → -234，对应 X 轴的力为-0.234N

0x01 0xFA 0xFF 0xFF: Fy 数据 0xFFFFFA01 → -1535，对应 Y 轴的力为-1.535N

0xEF 0x02 0x00 0x00: Fz 数据 0x000002EF → 751，对应 Z 轴的力为 0.751N

0x06 0x00 0x00 0x00: Mx 数据 0x00000006 → 6，对应 X 轴的力矩为 0.006Nm

0x0A 0x00 0x00 0x00: My 数据 0x0000000A → 10，对应 Y 轴的力矩为 0.010Nm

0x0F 0x00 0x00 0x00 Mz 数据 0x0000000F → 15，对应 Z 轴的力矩为 0.015Nm

0x6F: CRC16-CCITT 校验低字节

0x58: CRC16-CCITT 校验高字节

0x6F: 帧尾

0xF6: 帧尾

**注意：**

**传感器在上电后，需要预热一段时间以稳定输出，建议以最高的测量频率工作 10~20 分钟，使内部器件受热平衡后发送零点复位命令，复位完成后再开始使用。**

**命令#4 停止连续测量**

此命令将停止传感器的连续测量输出。

表 9. 停止连续测量命令

| 帧头 | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 |
|---|---|---|---|---|---|---|---|

| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
|---|---|---|---|---|---|---|---|---|---|
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x03 | 0xFF | 0xFC | 0x6F | 0xF6 |

返回数据：

表 10. 停止连续测量命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x03 | 命令字节 |
| 6 | 0x01 | 应答字节，成功为 0x01，失败为 0x00 |
| 7 | 0xB2 | CRC16-CCITT 校验低字节 |
| 8 | 0xC1 | CRC16-CCITT 校验高字节 |
| 9 | 0x6F | 帧尾第一字节 |
| 10 | 0xF6 | 帧尾第二字节 |

**注意：**

**如果传感器当前处于连续测量模式，在发送停止测量命令前，需要先发送 50 个字节的 0x00 占用 RS-485 总线，传感器收到这些字节后会暂时释放 RS-485 总线。主机在发送完 50 个字节的 0x00 数据后延时约 200ms 后发送停止测量命令即可正确停止传感器测量。传感器在接收到停止测量命令后，会返回一个应答数据包，表示已经正确停止了测量。如果主机在发送完 50 个字节的 0x00 数据后 250ms 内还未发送停止测量命令，则传感器会自动重新启动连续测量。如果传感器正处于连续测量模式下，在发送其他任何命令前，必须先停止连续测量，传感器在正确停止测量后才能正确接收其他的命令。**

**命令#5 设置测量频率**

此命令可设置传感器的连续测量频率，测量频率有 0~4 五个档位可以选择。

表 11. 设置测量频率命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | 测量频率 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 |
| 0xF6 | 0x6F | 0x04 | 0x00 | 0x00 | 0x05 | 0x00~ 0x04 | …… | …… | 0x6F | 0xF6 |

返回数据：

表 12. 设置测量频率命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x05 | 命令字节 |
| 6 | 0x01 | 应答字节，成功为 0x01，失败为 0x00 |
| 7 | 0x14 | CRC16-CCITT 校验低字节 |

| | | |
|---|---|---|
| 8 | 0x6B | CRC16-CCITT 校验高字节 |
| 9 | 0x6F | 帧尾第一字节 |
| 10 | 0xF6 | 帧尾第二字节 |

**命令#6 设置通讯波特率**

此命令可设置传感器的 RS-485 通讯波特率，传感器支持 9600bps, 115200bps, 128000bps, 256000bps, 460800bps, 600000bps, 750000bps, 921600bps 和 1500000bps 九种波特率，出厂默认波特率为 115200bps。该命令设置成功后立即生效，客户端需及时切换为新的波特率，否则将接收到错误的返回数据。

表 13. 设置通讯波特率命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | 波特率 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte6~9 | Byte10 | Byte11 | Byte12 | Byte13 |
| 0xF6 | 0x6F | 0x07 | 0x00 | 0x00 | 0x06 | …… | …… | …… | 0x6F | 0xF6 |

表 14. 不同波特率对应的 Byte6~9 的数值

| 波特率 | Byte6~9 的数值 | Byte10~11 的数值 |
|---|---|---|
| 9600 bps | 0x80, 0x25, 0x00, 0x00 | 0x45, 0x8C |
| 115200 bps | 0x00, 0xC2, 0x01, 0x00 | 0xED, 0x47 |
| 128000 bps | 0x00, 0xF4, 0x01, 0x00 | 0xE8, 0x30 |
| 256000 bps | 0x00, 0xE8, 0x03, 0x00 | 0x88, 0x60 |
| 460800 bps | 0x00, 0x08, 0x07, 0x00 | 0x7D, 0x0C |
| 600000 bps | 0xC0, 0x27, 0x09, 0x00 | 0x21, 0x36 |
| 750000 bps | 0xB0, 0x71, 0x0B, 0x00 | 0x58, 0xFE |
| 921600 bps | 0x00, 0x10, 0x0E, 0x00 | 0x27, 0x5C |
| 1500000 bps | 0x60, 0xE3, 0x16, 0x00 | 0x2D, 0x35 |

返回数据：

表 15. 设置通讯波特率命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x07 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x06 | 命令字节 |
| 6 | …… | 波特率第一字节 |
| 7 | …… | 波特率第二字节 |
| 8 | …… | 波特率第三字节 |
| 9 | …… | 波特率第四字节 |
| 10 | …… | CRC16-CCITT 校验低字节 |
| 11 | …… | CRC16-CCITT 校验高字节 |
| 12 | 0x6F | 帧尾第一字节 |
| 13 | 0xF6 | 帧尾第二字节 |

**命令#7 零点复位**

此命令用来复位传感器的零点输出，传感器安装完毕后发送该命令即可复位传感器的零点输出。

表 16. 零点复位命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x0B | 0xF7 | 0x7D | 0x6F | 0xF6 |

返回数据：

表 17. 零点复位命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x0B | 命令字节 |
| 6 | 0x01 | 应答字节，成功为 0x01，失败为 0x00 |
| 7 | 0x1B | CRC16-CCITT 校验低字节 |
| 8 | 0x48 | CRC16-CCITT 校验高字节 |
| 9 | 0x6F | 帧尾第一字节 |
| 10 | 0xF6 | 帧尾第二字节 |

**命令#8 修改设备地址**

此命令可用来修改传感器的设备地址，该功能使得在同一个 RS-485 总线上可使用多个传感器设备工作在单次测量模式下。

表 18. 修改设备地址命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | 新的设备地址 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 |
| 0xF6 | 0x6F | 0x04 | 0x00 | 0x00 | 0x0E | …… | …… | …… | 0x6F | 0xF6 |

返回数据：

表 19. 修改设备地址命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | …… | 新的设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x0E | 命令字节 |
| 6 | 0x01 | 应答字节，成功为 0x01，失败为 0x00 |
| 7 | …… | CRC16-CCITT 校验低字节 |
| 8 | …… | CRC16-CCITT 校验高字节 |
| 9 | 0x6F | 帧尾第一字节 |
| 10 | 0xF6 | 帧尾第二字节 |

**命令#9 复位设备地址**

当用户忘记设备设定地址时，使用此命令可将设备地址恢复为出厂设定值“0x00”。

表 20. 复位设备地址命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x0D | 0x31 | 0x1D | 0x6F | 0xF6 |

返回数据：

表 21. 复位设备地址命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x0D | 命令字节 |
| 6 | 0x01 | 应答字节，成功为 0x01，失败为 0x00 |
| 7 | 0xBD | CRC16-CCITT 校验低字节 |
| 8 | 0xE2 | CRC16-CCITT 校验高字节 |
| 9 | 0x6F | 帧尾第一字节 |
| 10 | 0xF6 | 帧尾第二字节 |

**命令#10 恢复用户设置**

此命令可用来从内部 Flash 中恢复用户保存的设置参数。设置参数恢复完成后注意设备地址和波特率有可能与当前设定不同。

表 22. 恢复用户设置命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x07 | 0x7B | 0xBC | 0x6F | 0xF6 |

返回数据：

表 23. 恢复用户设置命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x07 | 命令字节 |
| 6 | 0x01 | 应答字节，成功为 0x01，失败为 0x00 |
| 7 | 0x76 | CRC16-CCITT 校验低字节 |
| 8 | 0x0D | CRC16-CCITT 校验高字节 |
| 9 | 0x6F | 帧尾第一字节 |
| 10 | 0xF6 | 帧尾第二字节 |

**命令#11 恢复出厂设置**

此命令可用来从内部 Flash 中恢复出厂设置参数，包括：测量频率、通讯波特率、设备地址。

表 24. 恢复出厂设置命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x08 | 0x94 | 0x4D | 0x6F | 0xF6 |

返回数据：

表 25. 恢复出厂设置命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x08 | 命令字节 |
| 6 | 0x01 | 应答字节，成功为 0x01，失败为 0x00 |
| 7 | 0x48 | CRC16-CCITT 校验低字节 |
| 8 | 0x1D | CRC16-CCITT 校验高字节 |
| 9 | 0x6F | 帧尾第一字节 |
| 10 | 0xF6 | 帧尾第二字节 |

**命令#12 保存用户设置**

此命令可以将用户设置保存到传感器内部的 Flash 中，保存的用户设置断电不会丢失，并将在每一次传感器上电时被自动加载。

表 26. 保存用户设置命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x09 | 0xB5 | 0x5D | 0x6F | 0xF6 |

返回数据：

表 27. 保存用户设置命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x09 | 命令字节 |
| 6 | 0x01 | 应答字节，成功为 0x01，失败为 0x00 |
| 7 | 0x79 | CRC16-CCITT 校验低字节 |
| 8 | 0x2E | CRC16-CCITT 校验高字节 |
| 9 | 0x6F | 帧尾第一字节 |
| 10 | 0xF6 | 帧尾第二字节 |

**命令#13 获取传感器版本信息**

此命令可用来获取传感器的版本信息。

表 28. 获取传感器版本信息命令

| 帧头 | | 有效数据长度 | 设备地址 | 保留字节 | 命令 | CRC 低字节 | CRC 高字节 | 帧尾 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte 0 | Byte 1 | Byte2 | Byte3 | Byte4 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
| 0xF6 | 0x6F | 0x03 | 0x00 | 0x00 | 0x0A | 0xD6 | 0x6D | 0x6F | 0xF6 |

返回数据：

表 29. 保存用户设置命令的返回数据

| 字节号 | 数据 | 描述 |
|---|---|---|
| 0 | 0xF6 | 帧头第一字节 |
| 1 | 0x6F | 帧头第二字节 |
| 2 | 0x04 | 有效数据长度 |
| 3 | 0x00 | 设备地址 |
| 4 | 0x00 | 保留字节 |
| 5 | 0x0A | 命令字节 |
| 6 | …… | 固件生成时间，年 |
| 7 | …… | 固件生成时间，月 |
| 8 | …… | 固件生成时间，日 |
| 9 | …… | 主版本号 |
| 10 | …… | 小版本号 |
| 11 | …… | 修订版本号 |
| 12 | …… | CRC16-CCITT 校验低字节 |
| 13 | …… | CRC16-CCITT 校验高字节 |
| 14 | 0x6F | 帧尾第一字节 |
| 15 | 0xF6 | 帧尾第二字节 |

# 修订历史记录

表 30. 规格书修订历史记录

| Date | Revision | Description |
|---|---|---|
| 2019/09/20 | 2.0 | 将传感器规格和 RS485 接口命令说明分开 |
| 2019/10/14 | 2.1 | 修复获取设备 ID 命令 CRC16 计算错误问题 |

# 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.