



liangican

博客访问: 645440

博文数量: 6

博客积分: 0

博客等级: 民兵

技术积分: 95

用户组: 普通用户

注册时间: 2019-03-06 14:11

加关注

短消息

论坛

加好友

个人简介

我, 并不比别人差。

- 文章分类
- 全部博文 (6)

开黑库编译 (0)

嵌入式Linux (5)

Qt&Qt Creat (1)

未分配的博文 (0)


- 文章存档
- 2021年 (1)

2020年 (2)


2019年 (3)

我的朋友


最近访客




hhhhc5




wys2004




Mer_s




aabbme




killerp




jinrifen



TimesPa



zhangyuj



huazaich

- 推荐博文
- 学习Pytorch+Python之MNIST手...

·Linux 2.6.34 内核启动详细分...

·flowable 整合 springboot

·主备库心跳诊断 Heartbeat...

·MRP的各种问题及处理

- 相关博文
- 启明云端分享: ESP32-C3环境...

·启明云端分享: ESP32-C3环境...

·“出圈”易开发的智能86盒有...

·启明云端分享: ESP32-C3 系统...

·启明云端分享: esp32c3阿里云...

·启明云端分享: 该模块可满足...

·启明云端分享: 低功耗高性能...

·启明云端分享: 超详细ESP8266...

·启明云端分享: 86盒智能新型...

·高性能的机器学习让边缘计算...

推荐文章

Linux平台RS232/485串口编程实例

原创 分类: C/C++ 2020-03-08 16:16:07

在Linux下, 串口的读写跟文件的读写无异, 我们只需对相应的设备文件操作, 即可实现对串口的通讯. 这里给出的是一个实例, 具体概念的东西可能不会详细解释, 可自行百度. 简单来说串口通讯就是双方按照一定的数据格式发送接收数据, 一般是主从模式, 即主机发请求数据, 从机收到后返回对应的数据.

串口通讯的应用场景非常广泛, 常见的温湿度采集、自动门的控制等等. 因为需要对这些简单的装置信息采集或控制, 从而构建出一个综合的系统, 这里串口通讯必不可少, 方便、廉价.

下面就以温湿度采集作为实例写一篇博文.

我手上的这款温湿度是上海拓福电气SZ-WS系列温湿度变送器, 如下图: (大家不用纠结报文格式含义, 弄懂通讯原理即可举一反三)

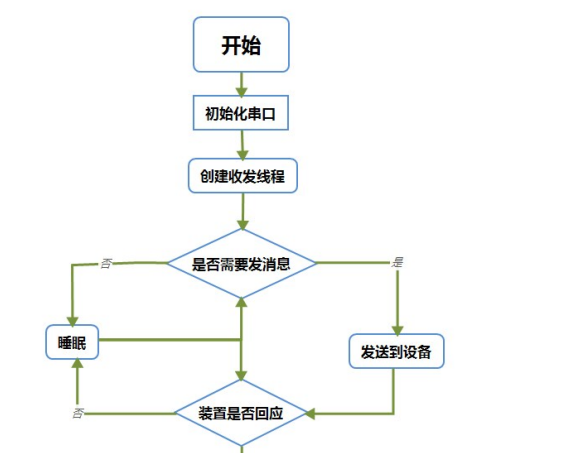


其中说明书主要是说了通讯规约, 即报文的格式: 如下

```

/*****
* 传感器->主站RS485帧结构
*
* | DestAddr 1Byte | MSG_TYPE 1Byte | DataLen 1Byte | Data <= 255Byte | CRC 2Byte |
*
*
*
*
*
* 主站->传感器 RS485帧结构
*
* | DestAddr 1Byte | MSG_TYPE 1Byte | star add 2Byte | Register Num 2Byte | CRC 2Byte
*
*
*
*
*
*****/
```

程序的流程大概如下, 没有消息发送和无数接收时都是睡眠状态, 释放CPU。



处理接收到的数据

部分代码解析如下：

main函数主要是创建温湿度类，然后0.5秒获取一次值，将其打印出来。

其中串口的参数要根据具体的设备来，ty设备就是对应的串口文件，具体怎么找出使用的串口是哪个ty这里就不讲解了，可自行百度。

```

1.  节点(此处)前叠或打开
2.  int main()
3.  {
4.      SERIAL_S stSerialParam;
5.      stSerialParam.u8BaudRate = BR_9600;
6.      stSerialParam.u8DataBit = 0; // 8bit
7.      stSerialParam.u8StopBit = 0; // 1bit
8.      stSerialParam.u8Check = 0; // None
9.      CHumitureManager *m_pCHumiture = new CHumitureManager("/dev/ttyS2", &stSerialParam, 1)
10. ;
11. while(1)
12. {
13.     printf("\033[0;31m [%s][%d] humidity=%d, temperature=%f\033[0;39m \n", __func__, __
14. _LINE__, m_pCHumiture->shumidity(), m_pCHumiture->temperature());
15.     mSleep(500);
16. }
17. }
```

温湿度管理模块的构造函数：

主要功能是根据传进来的参数初始化串口、创建读写和数据发送线程。

```

1.  节点(此处)前叠或打开
2.  CHumitureManager::CHumitureManager(const WD_C8 *pTtyDevPath, const SERIAL_S *pstSerialPara
3.  , WD_U8 SensorAdd) :
4.      m_FrtTemper(0), m_u32Rthumidity(0), m_enBaudRate(BR_9600)
5.  {
6.      assert(pTtyDevPath != NULL && pstSerialPara != NULL);
7.      m_u8SensorAdd = SensorAdd;
8.      /* 初始化串口并连接 */
9.      m_pCUart = new CUartOperator();
10.     m_pCUart->init(pTtyDevPath, pstSerialPara);
11.     CreateNormalThread(SendMsgThread, this, NULL);
12.     CreateNormalThread(ReceMsgThread, this, NULL);
13.     CreateNormalThread(cycleGetDeviceParam, this, NULL);
14. }
15. }
```

串口初始化大体流程：

```

1.  节点(此处)前叠或打开
2.  /* *****
3.  * Function Name : init
4.  * Parameter : pTtyDevPath,串口所使用的tty设备绝对路径,如/dev/tty02
5.  * Description : 配置串口参数
6.  * Return Value : On success, 0 is returned.
7.  * On error, -1 is returned
8.  * Author : LiangLiCan
9.  * Created : 2019/06/26
10.  * *****
11.  WD_S32 CUartOperator::init(const WD_C8 *pTtyDevPath, const SERIAL_S *pstSerialPara)
12.  {
13.      if (NULL == pTtyDevPath || NULL == pstSerialPara)
14.      {
15.          printf("Get Null pointer, Check!!!\n");
16.          return WD_FAILURE;
17.      }
18.      /* O_NOCTTY : 表示当前进程不希望与终端关联 */
19.      m_s32DevFd = open(pTtyDevPath, O_RDWR | O_NOCTTY);
20.      if (m_s32DevFd < 0)
21.      {
22.          printf("Open dev %s fail! \n", pTtyDevPath);
23.          return WD_FAILURE;
24.      }
25.      printf("Open dev %s success! \n", pTtyDevPath);
26.      /* 先清空参数 */
27.      struct termios stOldParam;
28.      bzero(&stOldParam, sizeof(stOldParam));
29.      tcsetattr(m_s32DevFd, TCSANOW, &stOldParam);
30.      //设置波特率
31.      if (SetBaudRate(m_s32DevFd, (BAUD_RATE_E)pstSerialPara->u8BaudRate) != WD_SUCCESS)
32.      {
33.          printf("SetBaudRate(%d) fail! \n", pstSerialPara->u8BaudRate);
34.          close(m_s32DevFd);
35.          return WD_FAILURE;
36.      }
37.      //设置数据位
38.      if (SetDataBit(m_s32DevFd, pstSerialPara->u8DataBit) != WD_SUCCESS)
39.      {
40.          printf("SetDataBit fail! \n");
41.          close(m_s32DevFd);
42.          return WD_FAILURE;
43.      }
44.      // 设置校验位
45.      if (SetCheck(m_s32DevFd, pstSerialPara->u8Check) != WD_SUCCESS)
46.      {
47.          printf("SetCheck fail! \n");
48.          close(m_s32DevFd);
49.          return WD_FAILURE;
50.      }
51.      //停止位
52.      if (SetStopBit(m_s32DevFd, pstSerialPara->u8StopBit) != WD_SUCCESS)
53.      {
54.          printf("SetStopBit fail! \n");
55.          close(m_s32DevFd);
56.          return WD_FAILURE;
57.      }
58.      m_bIsInit = true;
59.      return WD_SUCCESS;
60.  }
61. }
```

构建一帧数据函数体并加入链表：

构建好后添加链表，并唤醒发送线程。实际应用中我们会再增加一个对外的接口，如sendMsg()。用于二次封装AddSendFrameToLst函数，在需要的时候再发送消息，该例子是直接用了个线程定时循环去获取温湿度。

```

1.  节点(此处)前叠或打开
2.  /* *****
3.  * Function Name : AddSendFrameToLst
4.  * Parameter : enAddr是寄存器地址，bWrites是否是写寄存器
5.  * Description : 构建一帧完整的485数据，包括地址、功能码、CRC的帧值，并加入到发送链表中去
6.  * Return Value : On success, 0 is returned.
7.  * On error, 错误返回非0。
8.  * Author : LiangLiCan
9.  * Created : 2019/11/26
10.  * *****
11.  WD_S32 CHumitureManager::AddSendFrameToLst(REGISTER_ADD_E enAddr, WD_U16 u16ReadNum/* = 0
12.  */ , bool bWrite/* = False */ , WD_U16 u16SetData/* = 0 */)
13.  {
14.      CObjectLock ObjLock(&m_MuteSendLock);
15.      WD_U8 *pNode = NULL;
16.      WD_U8 aFrameHead[8] = {0};
17.      WD_S32 ret = 0;
18.      WD_U16 u16Temp = enAddr;
19.      aFrameHead[SFI_DEST_ADDR] = m_u8SensorAdd; // 总线上的设备地址
20.      aFrameHead[SFI_MSG_TYPE] = bWrite ? MT_WRITE : MT_READ;
21.      ShortToChar(u16Temp, &aFrameHead[SFI_REG_ADDR], true);
22.      if(bwrite){
23.          ShortToChar(u16SetData, &aFrameHead[SFI_REG_PARAM], true);
24.      }
25.      else {
26.          ShortToChar(u16ReadNum, &aFrameHead[SFI_REG_PARAM], true);
27.      }
28.      // 填充CRC
29.      ShortToChar(createCrcCode(aFrameHead, 6), &aFrameHead[SFI_CRC]);
30.      pNode = (WD_U8 *)malloc(sizeof(aFrameHead)); /* 发送线程会free掉它 */
31.      memcpy(pNode, aFrameHead, sizeof(aFrameHead));
32.      /* 添加入发送的列表 */
33.      m_SendBufLock.Lock();
34.      if (m_pSendBufList.empty()){
35.          m_SendBufLock.Signal();
36.      }
37.      m_pSendBufList.push_back(pNode);
38.      m_SendBufLock.Unlock();
39.  }
```

```
40.         return ret;
41.     }
```

发送数据线程:

主要功能是从链表中取出一帧数据发送, 无数据可写时处于休眠状态。

[点击\(此处\)查看更多打开](#)

```
1.  WD_VOID CHumitureManager::SendMsgThreadBody()
2.  {
3.      prctl(PR_SET_NAME, (WD_U32 *)"HumiSend");
4.      WD_U8 *pu8SenBufNode = NULL;
5.      while(1)
6.      {
7.          /* 从消息队列中取出一条发送的消息 */
8.          m_SendBufLock.Lock();
9.          if (m_pSendBufList.empty()){
10.              m_SendBufLock.Wait();
11.          }
12.          pu8SenBufNode = m_pSendBufList.front();
13.          m_pSendBufList.pop_front();
14.          m_SendBufLock.Unlock();
15.
16.          m_pCUart->writeData(pu8SenBufNode, 8);
17.          delete pu8SenBufNode;
18.      }
19. }
```

接收数据线程体:

[点击\(此处\)查看更多打开](#)

```
1.  WD_VOID CHumitureManager::ReceMsgThreadBody()
2.  {
3.      prctl(PR_SET_NAME, (WD_U32 *)"HumiRece");
4.      WD_S32 readlen = 0;
5.      WD_S32 readCount = 0; /* 已读数据长度 */
6.      WD_S32 MaxBufLen = sizeof(WD_U8) * MAX_FRAME_LEN;
7.      m_pReadBuf = new WD_U8[MaxBufLen];
8.
9.      while(1)
10.     {
11.         if(!m_pCUart->dataAvailable(100)){ // 是否有数据可读
12.             continue;
13.         }
14.         memset(m_pReadBuf, 0, sizeof(MaxBufLen));
15.         readCount = 0;
16.         do{
17.             if(m_pCUart->dataAvailable(100))
18.             {
19.                 readlen = m_pCUart->readData(m_pReadBuf + readCount, MaxBufLen);
20.                 if (readlen < 0){
21.                     break;
22.                 }
23.                 readCount += readlen;
24.             }
25.         }while(readCount < m_pReadBuf[RFI_DATA_LEN] + FRAME_EXTRA_LEN);
26.         // 处理读到的数据-----
27.         handleMsg(m_pReadBuf, readCount);
28.     }
29.     delete [] m_pReadBuf;
30. }
```

数据处理函数:

该函数在实际应用中也可以使用回调函数, 主要功能是处理拿到的数据。

[点击\(此处\)查看更多打开](#)

```
1.  WD_VOID CHumitureManager::handleMsg(WD_U8 *pMsgData, WD_U32 )
2.  {
3.      if(pMsgData[RFI_DEST_ADDR] != 0x1){// 判断有效性
4.          return ;
5.      }
6.
7.      if(pMsgData[RFI_MSG_TYPE] == MT_READ)
8.      {
9.          /* Baud Rate */
10.         m_enBaudRate = (BAUD_RATE_F)pMsgData[RFI_DATA + 1];
11.
12.         WD_U8 u8Backup = pMsgData[RFI_DATA + 2];
13.         /* 温度 */
14.         /* bit 15为温度正负值,0为正,1为负 */
15.         pMsgData[RFI_DATA + 2] &= 0x7f;
16.         m_fRtTemper = (u8Backup & 0x80 ? -CharToShort(&pMsgData[RFI_DATA + 2], true) : CharToShort(&pMsgData[RFI_DATA + 2], true)) / 10;
17.
18.         /* 湿度 */
19.         m_u32RtHumidity = CharToShort(&pMsgData[RFI_DATA + 4], true);
20.     }
21.     else if(pMsgData[RFI_MSG_TYPE] == MT_READ_ERR)
22.     {
23.         m_fRtTemper = 0;
24.         m_u32RtHumidity = 0;
25.         //DBG_HUMI_PRINT (LEVEL_ERROR, "Get Invalid read msg!\n");
26.     }
27.     //printf("\033[0;31m [%s] [%d]m_enBaudRate=%d Temper=%02f°C Humidity=%d\033[0;39m \n",
28.     _func_, __LINE__, m_enBaudRate, m_fRtTemper, m_u32RtHumidity);
}
```

大体的流程就上面了。具体的数据分析是根据具体的设备来的, 只需做下简单的修改即可移植到工程中来, 主要需要配置两点, 一是串口**通讯参数**和**tty设备**, 二是**帧结构**。流程和方法都是一样的, 以上例程供大家参考和学习, 有疑问欢迎一起留言交流

源码下载地址:

GitHub:

百度云:, 提取码ufk

源码下载下来直接make即可。需要交叉编译的修改一下编译选项, Makefile是通用模板, 修改很方便。

阅读(381662) | 评论(0) | 转发(1)

上一篇: AM335X修改nand分区大小或增加新分区
下一篇: 通用Makefile模板 (包含动态库和静态库)

0 赞

给主人留下些什么吧！~