|  |  |  |
| --- | --- | --- |
| 学号：XXXXXX |  | |
|  | | |
| 河南大学2017届本科毕业论文 | | |
| 基于STM32的液位监测报警装置及其上位机的制作 | | |
|  | | |
|  | | |
| 论文作者姓名： | X |  |
| 作 者 学 号： | X |  |
| 所 在 学 院： | X |  |
| 所 学 专 业： | 测控技术与仪器 |  |
| 导 师 姓 名： | X |  |
| 导 师 职 称： | 讲师 |  |
|  | | |
|  | | |
| **2017年 5月 15 日** | | |

摘要

在工业控制中，常常需要监测液面的高度来采取一定的控制措施，当液面高度超过或者低于某个阈值的时候自动采取某种措施，所以必须要有一个自动化的设备来进行监测并分析数据。本设计的主要功能是采用STM32单片机制作一个下位机，下位机接收传感器发送的信号并将其以一定的格式转发给上位机，上位机接收这个信号并将其显示和记录下来。

**关键词**：STM32，液位监测，上位机

Abstract:

In industrial control, often need to monitor the level of high to take control measures when the liquid height exceeds or falls below a certain threshold automatically to take certain measures, so there must be a monitoring and automated equipment to analyze the data. This design is mainly the STM32 microcontroller to make a slave, slave receiver sensor sends the signal and forwards it to a certain format to PC, PC receives the signal and displayed and recorded.

**Keywords**: STM32,Liquid level,Monitoring PC

目录

[摘要 I](#_Toc483503663)

[Abstract: II](#_Toc483503664)

[目录 III](#_Toc483503665)

[第一章 绪论 1](#_Toc483503666)

[第二章 整体设计 2](#_Toc483503667)

[2.1 总体框架 2](#_Toc483503668)

[2.2 器件选型 2](#_Toc483503669)

[2.2.1 传感器 2](#_Toc483503670)

[2.2.2 MCU 3](#_Toc483503671)

[2.2.3 上位机 3](#_Toc483503672)

[第三章 传感器 4](#_Toc483503673)

[3.1 原理 4](#_Toc483503674)

[3.2 电路 4](#_Toc483503675)

[第四章 下位机 6](#_Toc483503676)

[4.1 系统框图 6](#_Toc483503677)

[4.2 下位机主界面 7](#_Toc483503678)

[4.3 主要功能 7](#_Toc483503679)

[4.3.1 数据报告 7](#_Toc483503680)

[4.3.2 报警功能 8](#_Toc483503681)

[4.3.3 连接检测 8](#_Toc483503682)

[4.3.4 时钟校准 8](#_Toc483503683)

[4.3.5 回环测试 8](#_Toc483503684)

[4.3.6 警报解除 8](#_Toc483503685)

[4.3.7 操作显示 8](#_Toc483503686)

[第五章 上位机 9](#_Toc483503687)

[5.1 程序主界面 9](#_Toc483503688)

[5.2 类图 10](#_Toc483503689)

[5.2.1 字段 10](#_Toc483503690)

[5.2.2 属性和方法 11](#_Toc483503691)

[5.3 自定义类 12](#_Toc483503692)

[5.4 主要功能 13](#_Toc483503693)

[5.4.1 源数据显示与保存 13](#_Toc483503694)

[5.4.2 趋势图显示 14](#_Toc483503695)

[5.4.3 柱状图显示 14](#_Toc483503696)

[5.4.4 报警功能 14](#_Toc483503697)

[5.4.5 发送文字信息功能 14](#_Toc483503698)

[5.4.6 回复消息 15](#_Toc483503699)

[第六章 通讯 16](#_Toc483503700)

[6.1 通信方式 16](#_Toc483503701)

[6.2 通讯协议 16](#_Toc483503702)

[6.2.1 启动信号 16](#_Toc483503703)

[6.2.2 数据报告 16](#_Toc483503704)

[6.2.3 连接心跳信号 17](#_Toc483503705)

[6.2.4 时钟校准 17](#_Toc483503706)

[6.2.5 回环测试 17](#_Toc483503707)

[6.2.6 时间信息 17](#_Toc483503708)

[6.2.7 警报信息 17](#_Toc483503709)

[6.2.8 文字信息 18](#_Toc483503710)

[第七章 电路设计 19](#_Toc483503711)

[7.1 原理图 19](#_Toc483503712)

[7.2 PCB设计 19](#_Toc483503713)

[结论 21](#_Toc483503714)

[参考文献： 22](#_Toc483503715)

# 绪论

2013年中石化黄淮输油管道泄漏发生爆炸，2015年8月12日天津瑞海公司危险品仓库发生爆炸，2017年2月8日安徽铜陵化工厂爆炸，2017年3月11日新乡六通化工厂发生爆炸。这些事故频频发生让我们必须重视起化工厂的安全。化工行业常常使用液态试剂，设备运行时要实时监测试剂的液面高度，所以一套能自动检测高度并报警的设备是重要且极其有用的。

在学术网站上搜索“液面监测”可以得到约6万条结果，其中在临床医学领域1760条，化学领域1639条，石油与天然气领域1457条，所以对液面监测装置的研究有很多。但是大多数研究重点在于单独的设备，现在国家提倡“互联网+”，如果能让检测设备连接上电脑，就可以将监测结果发送至服务器存储起来，存储的数据可以被大数据技术利用。同时一些小工厂无力负担全套的自动检测设备。如果能有一种低成本的检测方案就可以使小工厂也能提高生产安全性。

# 整体设计

## 总体框架

使用购买的与被检测液体不会发生反应的传感器检测水位数据，之后将数据转换为电信号，MCU采用内置的ADC将电信号转化为数字信号，之后在MCU中通过计算得到实际水位的值。MCU将处理后的数据发送给上位机，上位机以图形界面显示出数据，并与设定阈值做比较以确定是否报警[1]。

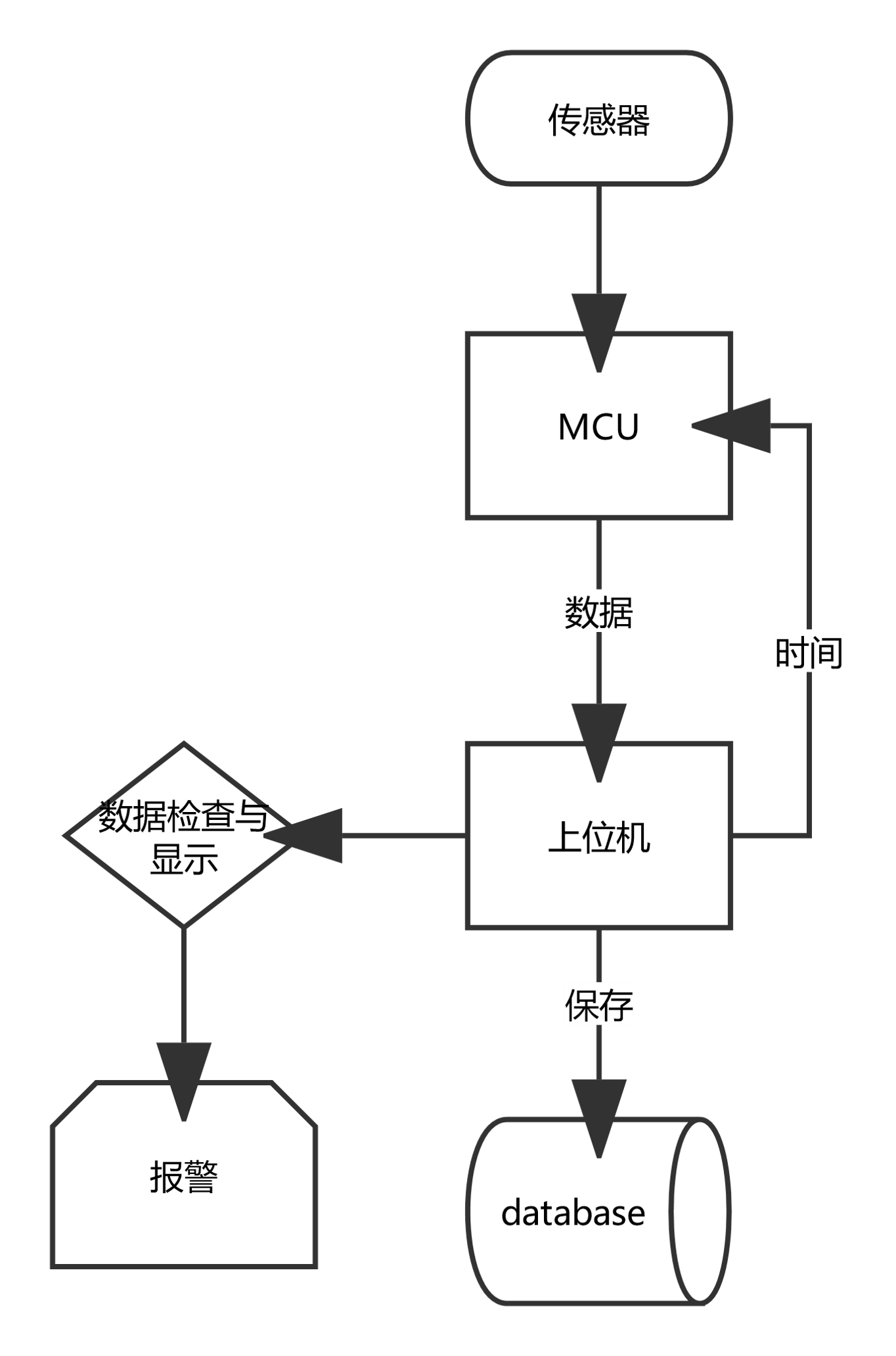


图2.1 总体框架图

## 器件选型

### 传感器

由于检测设备的目的是工业用途，而工业上用的液体种类繁多且有可能带有腐蚀性，所以不能设计一种通用的传感器，只能根据现场要求购买合适的不与被检测液体发生反应的传感器。常用的液位传感器输出均为电信号，并已经是成熟产品。如果实验中用水当作被测液体易造成短路等危险，且实验器材体积过大，操作不便。所以液位传感器使用一个电位器代替[2]。

### MCU

为了上位机的显示效果较好，所以数据报告的速率要求较高。每次报告数据均需要采集一次ADC的值，所以需要一个速度较快且具有硬件ADC的MCU。

STM32系列基于专为要求高性能、低成本、低功耗的嵌入式应用专门设计的ARM Cortex-M3内核[3]。按照内核架构的不同划分除了许多不同的系列。其中STM32F系列有： 101的“基本型”，103的“增强型”，105、107“互联型”，我们采用增强型的芯片STM32F103RBT6，其时钟频率高达72MHz，在同类产品中最高。虽然芯片只有44个引脚，但是他仍然具有丰富的外设接口，并且可以通过引脚复用功能充分利用引脚资源。所以我们采用STM32F103芯片作为工业现场的控制器。

### 上位机

工业现场主要使用的都是Windows系统，.NET Framework 4.0是支持生成和运行下一代应用程序和 XML Web Services 的内部 Windows 组件，采用C#语言编写[4]。。NET程序可以很方便的运行在各种版本的Windows系统上。所以我们选用C#作为上位机的开发语言。

# 传感器

液位传感器是一种测量液位的压力传感器。静压投入式液位计是基于所测液体静压与该液体的高度成比例的原理，使用隔离型扩散硅敏感元件或者陶瓷电容压力敏感传感器，将静压转换为电信号，再经过温度补偿与线性修正将其转化为标准电信号（一般为4～20mA/1～5VDC）[5]。

## 原理

用静压测量原理：当把液位传感器投入到被测液体中某一深度时，传感器迎液面受到的压强公式为[6]：

（3‑1）

之后可以采用压力传感器将压力信号转化为电信号，常用的压力传感器是利用半导体单晶硅的压阻效应制成的一种敏感元件，又称半导体应变片[7]。压阻效应是指半导体晶体材料在某一方向上受力后，半导体材料会产生变形，材料的电阻率会因为形变的关系发生变化。半导体应变片需要粘贴在试件上测量试件形状的改变，或者粘贴在弹性敏感元件上间接地感受被测外力，采用不同构造，形状的弹性敏感元件可测量不同物体的应力、应变、压力、扭矩、加速度等机械量[8]。

## 电路

由于液位传感器输出的电信号主要为电压信号。为了安全与操作方便，所以我们可以采用电位器来代替传感器，本设计所用开发板的电位器原理图如下：

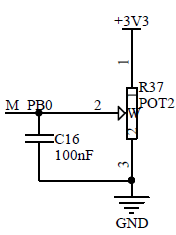


图3.1 电位器

本电位器的最大电压正好为MCU的量程，所以不需要做额外的处理，但是当采购的传感器输出电压不是在0V~3.3V之间时，我们可以用比例放大电路将电压信号归一化为0V~3.3V，这样即可使用STM32的内置ADC将其转化为数字信号。

常用电压放大电路



图3.2 电压放大电路

其电压关系为

 （3‑2）

无源I-V变换电路

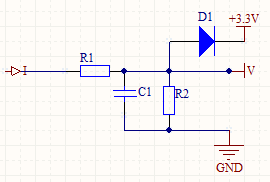


图3.3 无源I-V变换电路

其电压关系为

（3‑3）

# 下位机

开发板采用国信长天公司的CT117E型号开发板，此开发板采用STM32F103RBT6单片机，有7个LED，一个蜂鸣器，一个显示器作为输出设备。有4个按键，一个电位器作为输入设备。有2K的EEPROM和SPI通信的读卡器。采用串口与上位机通信。

CT117E开发板采用串口的方式下载程序，安装必要的驱动后可以使用Keil5的在线调试功能，为程序的调试提供方便。

CT117E开发板采用USB 5V供电，内部通过AMS1117芯片将电压降至3.3V供给单片机和其他外设。

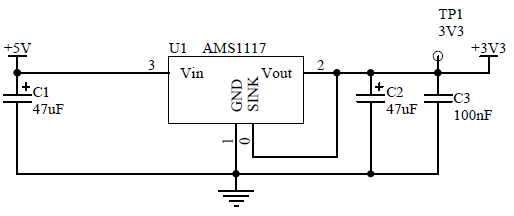


图4.1 降压电路

由于EEPROM的速度慢，内存卡硬件复杂易损坏，所以数据保存工作交给上位机，下位机只负责数据的采集与处理。

下位机的开发环境为Keil Software 软件公司的Keil5 ARM版，开发语言为语言为C99标准的C语言。

## 系统框图

本设备采用定时报告的方式向上位机发送数据，定时周期由任务需求决定。为了上位机的显示效果较好，我们采用了10ms报告一次数据。

SysTick\_Config(SystemCoreClock/1000);

上条语句将系统时钟配置为1ms一次中断，在SysTick\_Handler()函数中设定计数变量，当计数变量对10进行取余运算结果为0的时候执行AD转换并向上位机报告。

设备的主要功能是在中断中完成的，在主函数中只做了初始化工作。主循环内采用按键扫描的方式检测按键指令。中断系统的系统框图如下：

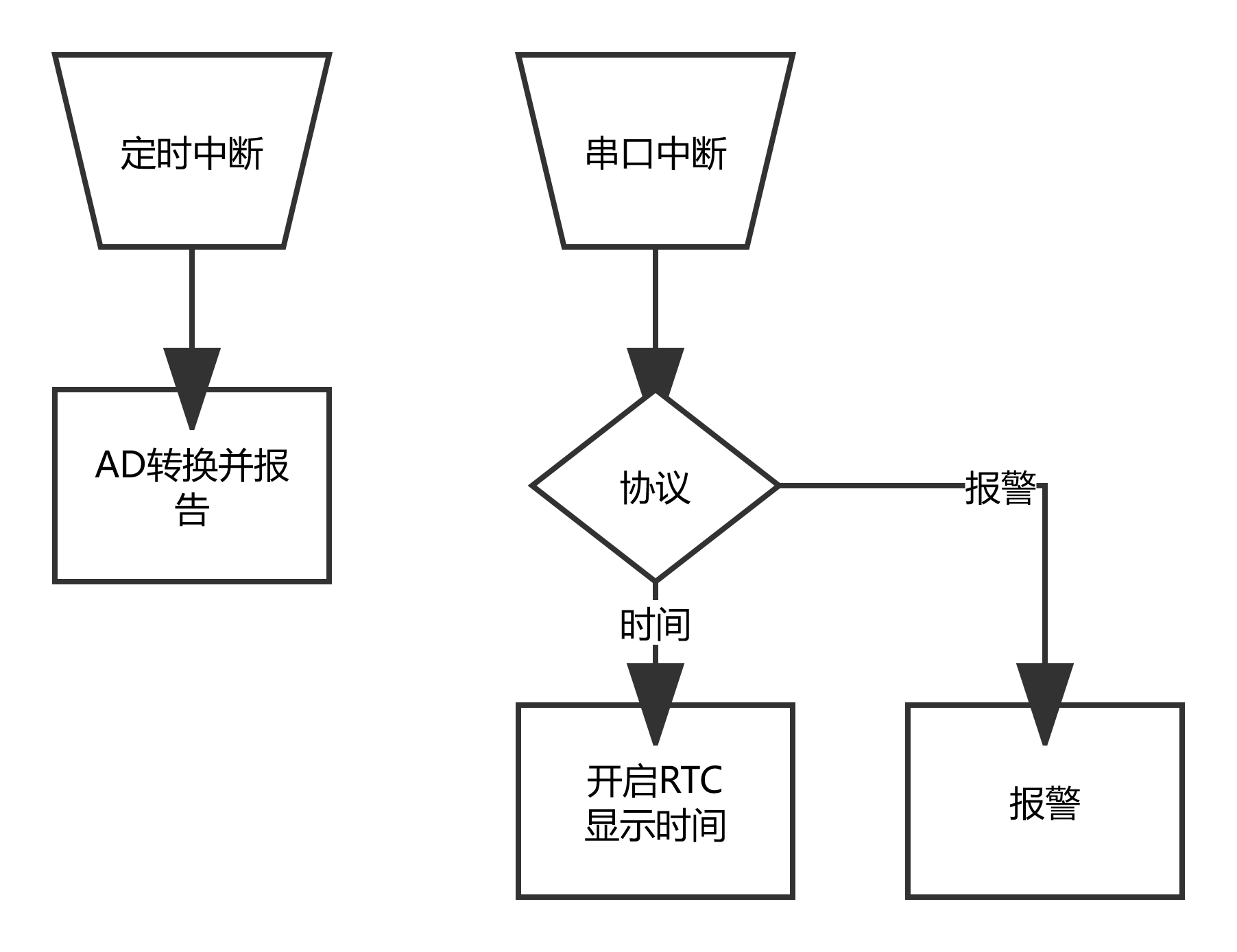


图4.2 下位机系统框图

## 下位机主界面

下位机配备一块320\*240像素的TFT显示屏，采用横屏的方式显示10行信息，每行高24像素。第一行显示设备名，第二行显示高度，第三行显示读取到的ADC的电压。第七行是当前时间，在单片机与上位机连接后获取上位机时间开始显示。第八行显示连接状态与操作指令。最后一行显示上位机发送的信息。当上位机有信息发送过来的时候，高度、ADC、时间的显示暂时停止，为接收图像信息留出足够的系统资源，当接收完毕时，所有数据恢复正常，时钟会将接收所花费时间跳过去，不会因为接收数据造成时钟不准确。



图4.3 下位机主界面

## 主要功能

### 数据报告

每过10ms，下位机向上位机报告一次液位信息。下位机不进行超限判断，也不存储阈值信息。

### 报警功能

当上位机接收到的高度信息超过或者低于阈值的时候将发出警报，同时通知下位机也发出警报。下位机接收到信号后需要开启蜂鸣器，同时LED灯以1Hz的频率闪烁。

### 连接检测

下位机必须要保证和上位机处于连接状态，当连接断开超过6秒钟时，下位机发出报警信号。由于下位机不保存阈值信息，此时需要人为的通过显示屏上的高度判断是否正常，等待上位机的重新连接。

### 时钟校准

当需要主动校准时钟时，按下按键1则通知上位机向下位机发送一次时间信息。

### 回环测试

下位机可以主动向上位机发送回环测试请求，上位机接收到后以报警信号回应，下位机成功接收后以报警状态显示。此功能主要用于手动判断与上位机的连接状况。

### 警报解除

当下位机发起回环测试的时候会进入报警状态，此时可以通过按键3解除警报。但是当上位机判断液位超过或者低于阈值的时候无法解除报警。连接断开的报警也无法解除。

### 操作显示

当进行时钟校准，回环测试，警报解除的时候，下位机要在第八行显示操作指令两秒钟，两秒后切换回连接状态指示。

# 上位机

上位机采用C#语言编写，是一个基于.NET框架的Winform应用程序，开发环境为Visual Studio 2015，采用.NET4.0和C#6标准。

C#是一种安全的、稳定的、简单的、优雅的由C和C++衍生出来的面向对象的编程语言，它在继承C和C++强大功能与高运行效率的同时去掉了包括宏、多重继承等复杂功能，使得程序的编写更加优雅[8]。C#有着一个强大的IDE——Visual Studio。在Visual Studio中，我们可以用可视化的方式编写Windows应用程序，同时我们还可以使用.NET提供的一系列工具和服务来最大限度的减小开发难度。

## 程序主界面

程序的主界面分为左右下三个部分，分别有着不同的功能。

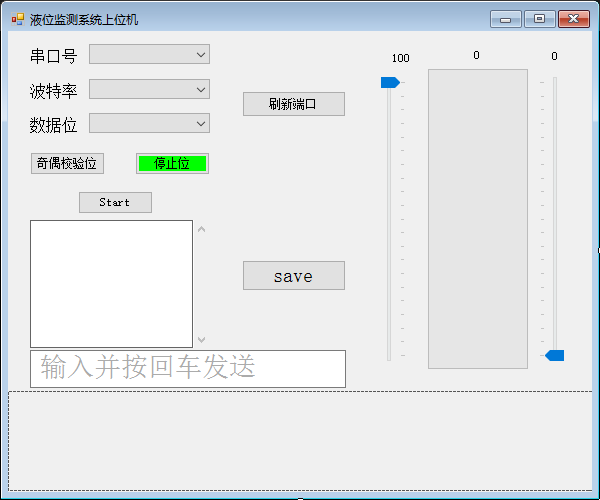


图5.1 程序主界面

左侧上部为控制区。当程序启动的时候，首先检查串口并将其存入到一个字符串数组中，当没有串口设备处于连接状态时显示错误信息，同时Start按钮显示为“No Link”，按钮为不可用状态。如果有串口则选择默认串口号，波特率，数据位。右侧为刷新串口，当无串口或者需要更新串口的时候点击。

“串口号”列表内显示的有当前所有链接的的串口设备，默认为最后一个串口设备。“波特率”列表内有4800、9600、19200、115200、四种选项，默认为115200。数据位有5、6、7、8、四种，默认为8位。“奇偶校验位”仅支持奇校验或无校验，默认为无校验。停止位仅支持一位停止位或无停止位，默认为一位。

左侧下部为数据区，当点击Start之后用控制区内选择的数据打开串口。Start下面为接收到的数据的显示区域。右侧为数据保存按钮。接收框下侧为发送框，可以发送不超过12个全角字符或者22个半角字符文字给下位机。

右侧是阈值设定界面和高度的显示柱状图，左侧为最大阈值，右侧为最小阈值。当液位高于左侧或者低于右侧的时候系统报警。

程序界面最下端是趋势图，根据接收到的数据绘制液位变化趋势。

## 类图

C#是一个面向对象的语言，所以主函数只有三句话。

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

所有的功能全部通过Form1这个类中的方法实现，所以程序的主要内容都在这个类中。类图(Class diagram)以图形的方式显示了模型的静态结构，所以我们可以用类图展示出类的内部结构。类图不显示暂时性信息。

### 字段

字段(field)是类的一个成员。是表示与对象或类关联的变量。

字段中灰色部分为系统控件名称，黑色部分为全局公用字段。lastCilck和mouseOffset是用于悬浮窗的变量，pointStart，pointValue，pointEnd三个变量是底部panel绘图使用变量，trimChar为通讯协议头，用于在处理数据的时候将协议头去掉。warring是属性用的全局变量。

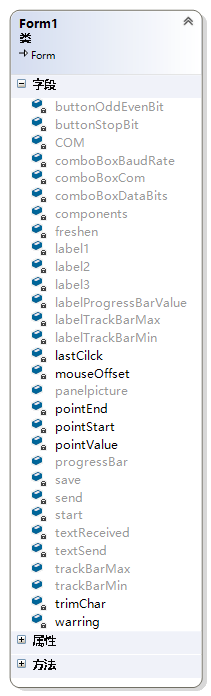


图5.2 字段

### 属性和方法

属性是对字段的封装，属性的实质是方法（get方法和/或set方法）[8]。属性既然是方法，那速度就一定比字段要慢，但是字段不能检查变量是否有效，也不能在变量被改变时及时的发出信息。所以采用属性将字段封装起来，这样就可以在每次改变字段的值的时候检查是否需要发出警报信号。

程序中仅使用了一个属性，用于在数据超过或者低于阈值的时候发出警报声音和向下位机传送警报信息。

C#中的方法即是C语言中的函数，是个用于执行某些操作的集合。

灰色的方法为软件开启时的系统初始化及软件关闭时资源释放。其余的方法大多是对用户操作的响应，其中COM\_DataReceived函数是串口接受到的事件函数，hideForm1和showForm1是用于打开和关闭悬浮窗时对于窗口的处理函数，warringChange函数是当Warring属性改变的时候调用的报警函数。SendTime函数是向下位机发送当前时间的函数。

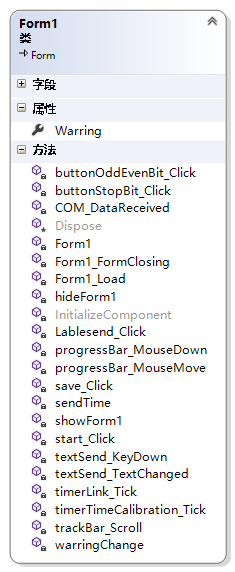


图5.3 属性和方法

## 自定义类

柱状图的显示采用System.Windows.Forms.ProgressBar类，但是这个类默认只能水平放置，所以要重写构造参数实现垂直的进度条。

public partial class VerticalProgressBar : ProgressBar

{

protected override CreateParams CreateParams

{

get

{

CreateParams cp = base.CreateParams;

cp.Style |= 0x04;

return cp;

}

}

}

因为下位机屏幕大小有限，且受到数据传送速率限制，所以不能发送过多数据给下位机，所以发送文本的长度需要受到限制。可是TextLength函数在计算文本长度的时候不会区分中文和英文，他会将中文和英文都按照一个字符计算。可是在下位机的显示上，中文的宽度是英文的两倍。所以需要有一个可以区分中英文的函数，这个函数的目的是限制文本框内容的视觉宽度，实质是检查内容的字节数。于是我们采用重写OnKeyPress函数的方法使得超出视觉范围的字符被截取掉。

public partial class SmartTextBox : TextBox

{

uint \_maxByteLength = 0;

public uint MaxByteLength

{

get { return \_maxByteLength; }

set { \_maxByteLength = value; }

}

protected override void OnKeyPress(KeyPressEventArgs e)

{

base.OnKeyPress(e);

if (ReadOnly) return;

if (\_maxByteLength == 0) return;

if (char.IsControl(e.KeyChar)) return;

int textByteLength = Encoding.GetEncoding(950).GetByteCount(Text + e.KeyChar.ToString());

int selectTextByteLength = Encoding.GetEncoding(950).GetByteCount(SelectedText);

if (textByteLength - selectTextByteLength > \_maxByteLength) e.Handled = true;

}

}

## 主要功能

### 源数据显示与保存

下位机将数据以一定协议发送给上位机，上位机将接收到的数据在左侧接受框内显示出来。当点击右侧保存按钮时，数据将被存储在程序所在目录下的“rec.txt”文件内，并记录保存时间。同时接收框里的内容会被清空。

当接收到错误的数据时，比如串口关闭时数据被截断，串口开启时数据只接收到后半段，程序要识别出错误的数据并显示错误的信息，同时将错误数据丢弃不显示在柱状图中。

### 趋势图显示

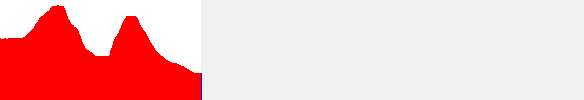


图5.4 趋势图

当接收到下位机传送的数据时，最下端会自动绘制当前页面高度。随着数据的增多，就会连接成表示液位变化的趋势图。

注：为了降低上位机的性能占用，Graphics对象采用临时方式绘制，减小实时更新bmp文件头的性能开销。但是这样会造成当窗口移动到屏幕外时，已绘制对象被释放。考虑到有悬浮窗功能，不需要移动窗口到边缘，所以采用了这个具有缺陷的方法。

### 柱状图显示

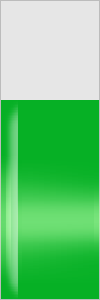


图5.5 柱状图

程序的右半部分是柱状图显示界面，此处可以以图形的方式将最后一次接收到的数据显示在柱状图中，柱状图的最大值为100，空值为0，当双击柱状图时会以悬浮窗的状态显示高度。再次双击可以显示原窗口。

### 报警功能

右侧的两个trackbar控制两个阈值，当液位大于左侧阈值或者低于右侧阈值的时候上位机发出声音警报。同时通知下位机也发出警报。

### 发送文字信息功能

上位机可以给下位机发送文字信息，最高发送12个全角字符或者24个半角字符。当发送的时候，下位机的时间显示与时间校准功能会被暂时关闭，等数据传输完毕的时候会自动打开。在时间显示被关闭的时候，内部时间正常走动，恢复显示时会将传输所用时间跳过。

### 回复消息

下位机会向上位机发送一些请求，比如时间校准，回环测试等。当上位机接受到这些请求的时候应该发送对应的回复信号。

# 通讯

## 通信方式

串行接口(Serial Interface)简称串口，也称串行通信接口或串行通讯接口，是采用串行通信方式的扩展接口，串行接口是指数据一位一位地顺序传送，其特点是通信线路简单，只要一对传输线就可以实现双向通信（可以直接利用电话线作为传输线），从而大大降低了成本，特别适用于远距离通信，所以常用于工业控制当中[11]。

串口通信有两种最基本的方式：同步串行通信方式和异步串行通信方式：

同步串行接口（英文：Synchronous Serial Interface，SSI）是一种常用的工业用通信接口[12]。但是同步串行接口需要同步时钟，所以我们不采用此种方案。异步串行是指UART（Universal Asynchronous Receiver/Transmitter），通用异步接收/发送，数据随时在需要的时候发送过去。

本设备与上位机之间采用异步串口通信。

我们采用STM32中的内置硬件串口设备，波特率可以任意调节。由于需要发送图形信息，所以本设备采用较为高速的115200。

上位机我们也可以选择任意的波特率，内部定义了4800，9600，19200，115200四种常见的波特率，默认与下位机匹配采用115200。

## 通讯协议

上位机与下位机的通信均通过串口进行，采用约定好的数据头表示此次通讯的目的，通讯头之后发送数据，采用回车符作为结束符，收到回车符后开始解析命令并执行。

具体的通讯内容如下（信号长度均不计算结束标识符“\n”）：

### 启动信号

方向：MCU🡺PC

长度：4字节。

通信协议： Boot

发送时间：MCU初始化后发送一次，重启后初始化也会发送。

send**(**"Boot"**);**

### 数据报告

方向：MCU🡺PC

长度：2~4字节。

通信协议： H+液面高度

发送时间：MCU初始化完成后即开始以100Hz的频率发送。

sprintf**(**ch**,**"H%d\n"**,**height**);**

send**(**ch**);**

### 连接心跳信号

方向：PC🡺MCU

长度：4字节。

通信协议： Link

发送时间：串口被打开后每过2秒钟发送一次。

COM.WriteLine("Link");

### 时钟校准

方向：MCU🡺PC

长度：4字节。

通信协议： Time

发送时间：按键1松开后。

send**(**"Time"**);**

### 回环测试

方向：MCU🡺PC

长度：5字节。

通信协议： Alarm

发送时间：按键2松开后。

send**(**"Alarm"**);**

### 时间信息

方向：PC🡺MCU

长度：16字节。

通信协议： T+时间；（yyyyMMdd hhmmss）

发送时间：上位机的串口成功打开时、接收到时钟校准信号时。

DateTime time **=** DateTime**.**Now**;**

time **=** time**.**AddSeconds**(**2**);**

COM**.**WriteLine**(**'T' **+** time**.**ToString**(**"yyyyMMdd HHmmss"**)** **+** '\n'**);**

### 警报信息

方向：PC🡺MCU

长度：3或4字节。

通信协议： 开启警报：Aon；

关闭警报Aoff；

if (COM.IsOpen)

{

if (value == true)

COM.WriteLine("Aon");

else

COM.WriteLine("Aoff");

}

### 文字信息

方向：PC🡺MCU

长度：2~6字节。

通讯协议： 开始传送：Pclear；

行信号：Py+行号；

字符：P1；

背景：P0；

结束传送：Pover；

COM.WriteLine("\r\n");

COM.WriteLine("Pclear");

COM.WriteLine("\r\n");

for (int y = 0; y < 24; y++)

{

COM.WriteLine("Py" + ((char)y).ToString());

for (int x = 0; x < 320; x++)

{

if (bmp.GetPixel(x, y).GetBrightness() < 0.8f)

COM.WriteLine("P1");

else

COM.WriteLine("P0");

}

}

COM.WriteLine("Pover");

# 电路设计

工业现场如果使用原开发板作为核心，则设备价格较高，且不容易做电磁屏蔽，所以需要简化电路，去掉无用的设备，并且增加电磁屏蔽罩的插槽，使得设备可以在有电磁干扰的环境下运行。

## 原理图

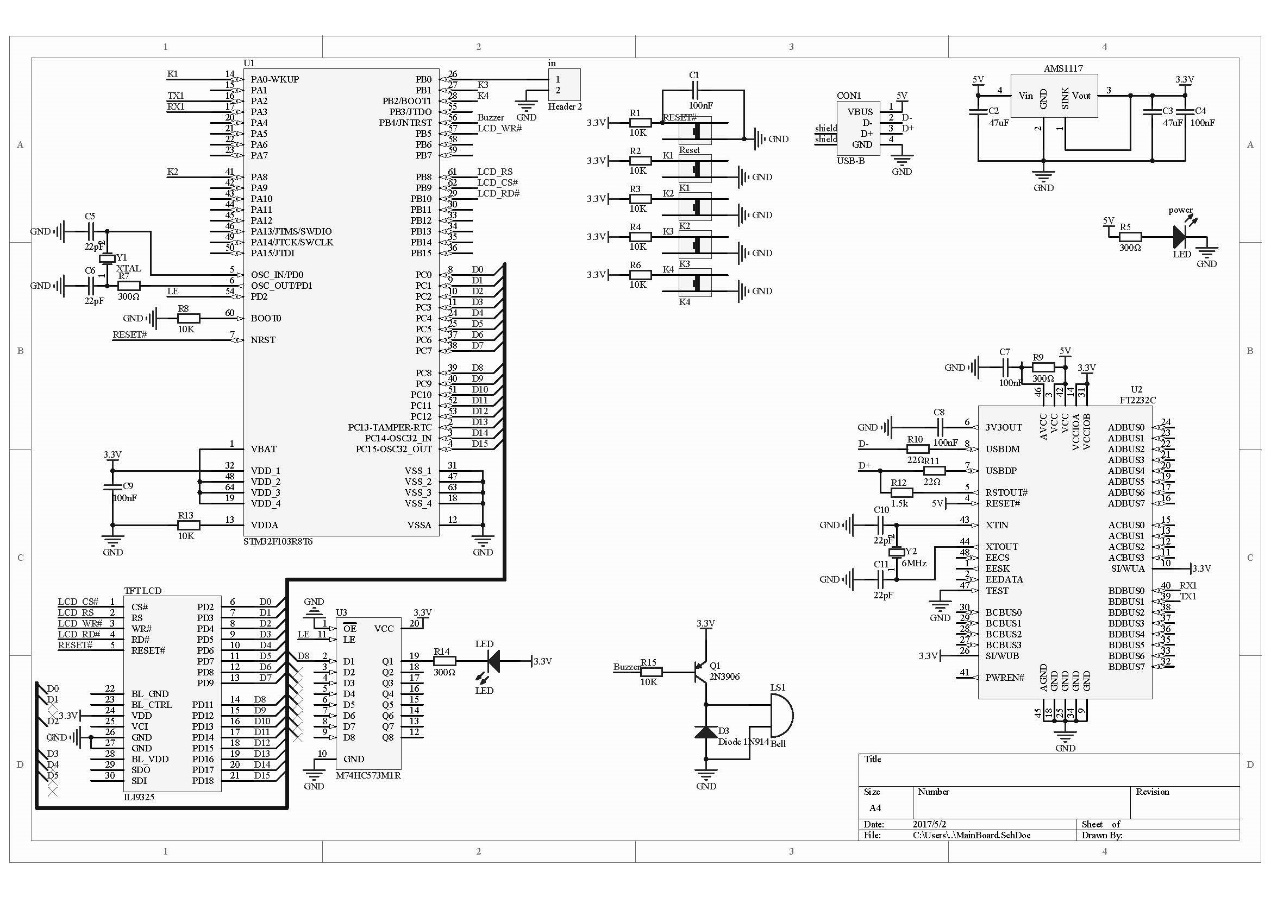


图7.1 原理图

## PCB设计

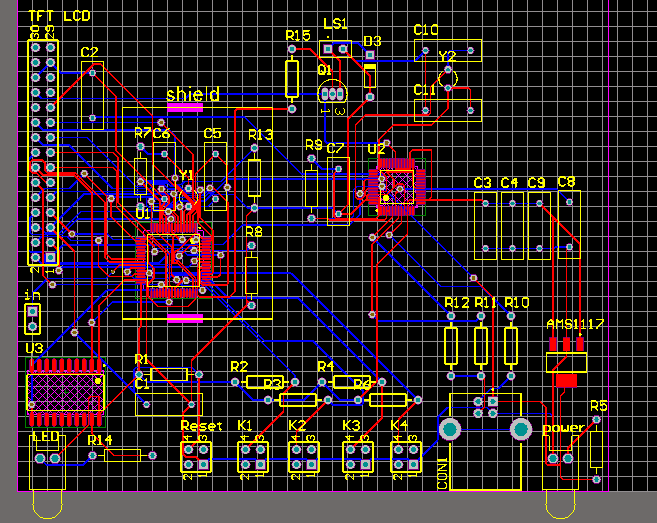


图7.2 PCB图

在电路的核心部分增加了屏蔽罩的插槽，可以将主芯片与晶振系统放入屏蔽罩内。由于设备不需要量产，所以除下集成电路外均采用了直插式元件。

# 结论

为了工业运行的安全，所以设计了这个设备，设备完成了从数据采集到数据处理与显示到数据的存储的所有功能。上位机可以稳定运行且能排除错误数据，具有长时间稳定运行的能力，可以胜任长时间的工业环境运行。同时设备的结构简单，只需要一个传感器，一块STM32的系统板和一台计算机即可运行。串口通信可以采用普通双绞线，设备部署成本低，工作简单。

由于下位机的性能问题，设备还是有一些缺陷的，比如晶振频率漂移造成时钟不准确需要经常校准，解决办法是上位机每过两个小时矫正一次下位机的时间，保证误差不超过一分钟，不影响正常使用。还可以采用主动校准的方式获得准确时间。

# 参考文献：

1. 杨朝虹, 李焕. 新型液位检测技术的现状与发展趋势[J]. 工矿自动化, 2009, 35(6):61-64.
2. 于治会. 液位传感器[J]. 传感器与微系统, 1990(1):41-45.
3. 田中宏明. 压力传感器:, CN 1892198 A[P]. 2007.
4. 汤同奎, 邵惠鹤. 一种基于半导体压力传感器的液位测量方法[J]. 自动化仪表, 1999, 20(3):9-10.
5. 付玉志, 麻红昭, 林晓伟. 基于STM32沼液液位监控系统的设计[J]. 工业控制计算机, 2014(6):149-150.
6. 王永虹, 徐炜, 郝立平. STM32系列ARM Cortex-M3微控制器原理与实践[M]. 北京航空航天大学出版社, 2008.
7. Ai D. .NET Framework Architecture[J]. Computer Engineering & Applications, 2003.
8. 钱哨, 李挥剑, 李继哲. C# WinForm实践开发教程[M]. 中国水利水电出版社, 2010.
9. 郭维芹. 模拟电子技术[M]. 科学出版社, 1993.
10. 郑彪, 汪秉文. 串口通信在工业控制中的应用[J]. 自动化仪表, 2002, 23(4):58-59.
11. 张雄飞, 方方. Windows平台下数据采集串口通讯的实现[J]. 计算机测量与控制, 2001, 9(3):66-68.

# 附录一

实物图



# 附录二

## 下位机源代码

void SysTick\_Handler(void)

{

ms++;

if(ms%10==0)

{

ADC\_SoftwareStartConvCmd(ADC1, ENABLE);

adv=ADC\_GetConversionValue(ADC1)\*3.3/0xfff;

height=adv\*100/3.3;

if(!displaystate)

{

sprintf(ch,"Height=%-3d",height);

LCD\_DisplayStringLine(Line1,(u8\*)ch);

sprintf(ch,"ADC:%1.2fV",adv);

LCD\_DisplayStringLine(Line2,(u8\*)ch);

}

sprintf(ch,"H%d",height);

send(ch);

}

if(ms%500==0)

{

link++;

if(warring==true)

{

ledstate^=256;

GPIOC->BSRR=ledstate;

GPIOD->BSRR=1<<2;

GPIOD->BRR=1<<2;

}

if(link>10)

{

LCD\_ClearLine(Line7);

LCD\_DisplayStringLine(Line7,(u8\*)"No link");

warring=true;

GPIO\_ResetBits(GPIOB,GPIO\_Pin\_4);

}

}

}

void USARTz\_IRQHandler(void)

{

if(USART\_GetITStatus(USARTz, USART\_IT\_RXNE) != RESET)

{

USART\_ClearITPendingBit(USARTz, USART\_IT\_RXNE);

USARTbuff[USARTcount]=USART\_ReceiveData(USARTz);

if(USARTbuff[USARTcount]==10)

{

USARTcount=0;

//Time

if(USARTbuff[0]=='T')

{

USART\_ITConfig(USART2,USART\_IT\_RXNE,DISABLE);

//string to int

yyyy=(USARTbuff[1]-'0')\*1000+(USARTbuff[2]-'0')\*100+(USARTbuff[3]-'0')\*10+USARTbuff[4]-'0';

MM=(USARTbuff[5]-'0')\*10+(USARTbuff[6]-'0');

dd=(USARTbuff[7]-'0')\*10+(USARTbuff[8]-'0');

hh=(USARTbuff[10]-'0')\*10+(USARTbuff[11]-'0');

mm=(USARTbuff[12]-'0')\*10+(USARTbuff[13]-'0');

ss=(USARTbuff[14]-'0')\*10+(USARTbuff[15]-'0');

//Display

sprintf(timedisplaybuff,"%d-%d-%d %2d:%2d:%2d",yyyy,MM,dd,hh,mm,ss);

LCD\_DisplayStringLine(Line6,(u8\*)timedisplaybuff);

//enable rtc

RTC\_SetCounter(hh\*3600+mm\*60+ss);

RTC\_WaitForLastTask();

//Enable the RTC Interrupt

NVIC\_RTC\_InitStructure.NVIC\_IRQChannel = RTC\_IRQn;

NVIC\_RTC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1;

NVIC\_RTC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

NVIC\_RTC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_RTC\_InitStructure);

USART\_ITConfig(USART2,USART\_IT\_RXNE,ENABLE);

}

//From point to graph

if(USARTbuff[0]=='P')

{

if(USARTbuff[1]=='c')//start

{

LCD\_ClearLine(Line9);

displaystate=true;

LCD\_ClearLine(Line7);

LCD\_DisplayStringLine(Line7,(u8\*)"Wait....");

return;

}

if(USARTbuff[1]=='o')//stop

{

displaystate=false;

return;

}

//ctrl

if(USARTbuff[1]=='y')

{

LCD\_SetCursor(215+(int)USARTbuff[2],0);

LCD\_WriteRAM\_Prepare();

return;

}

if(USARTbuff[1]=='1')

{

LCD\_WriteRAM(Red);

}

if(USARTbuff[1]=='0')

{

LCD\_WriteRAM(White);

}

}

//Alarm

if(USARTbuff[0]=='A')

{

if(USARTbuff[2]=='n')

{

warring=true;

GPIO\_ResetBits(GPIOB,GPIO\_Pin\_4);

}

else

{

warring=false;

ledclose();

GPIO\_SetBits(GPIOB,GPIO\_Pin\_4);

}

}

//Link

if(USARTbuff[0]=='L')

{

if(link>10)

{

warring=false;

ledclose();

}

link=0;

LCD\_ClearLine(Line7);

LCD\_DisplayStringLine(Line7,(u8\*)"Link");

}

}

else

{

USARTcount++;

}

}

}

void RTC\_IRQHandler(void)

{

if (RTC\_GetITStatus(RTC\_IT\_SEC) != RESET)

{

RTC\_ClearITPendingBit(RTC\_IT\_SEC);

RTC\_WaitForLastTask();

ss=RTC\_GetCounter();

/\* 23:59:59 \*/

if (ss >= 0x00015180)

{

RTC\_SetCounter(0x0);

RTC\_WaitForLastTask();

dd++;

}

hh = ss / 3600;

mm = (ss % 3600) / 60;

ss = (ss % 3600) % 60;

if(!displaystate)

{

LCD\_ClearLine(Line6);

sprintf(timedisplaybuff,"%d-%d-%d %2d:%2d:%2d",yyyy,MM,dd,hh,mm,ss);

LCD\_DisplayStringLine(Line6,(u8\*)timedisplaybuff);

}

}

}

## 上位机源代码

public partial class Form1 : Form

{

/// <summary>

/// 初始化

/// </summary>

public Form1()

{

InitializeComponent();

}

/// <summary>

/// 窗口载入

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form1\_Load(object sender, EventArgs e)

{

string[] comlist = SerialPort.GetPortNames();

comboBoxCom.Items.Clear();

if (comlist.Length == 0)

{

MessageBox.Show("无串口设备");

//freshen.Visible = true;

if (start.Text == "Start")

{

start.Text = "No Link";

start.Enabled = false;

COM.Close();

}

}

else

{

//freshen.Visible = false;

foreach (var s in comlist)

{

comboBoxCom.Items.Add(s);

}

//comboBoxCom选定条目

comboBoxCom.SelectedIndex = 0;

//comboBoxBaudRate 1.9600 3.38400 4.115200

comboBoxBaudRate.SelectedIndex = 4;

//comboBoxWordLength

comboBoxDataBits.SelectedIndex = 3;

//开始按钮使能

start.Enabled = true;

if (start.Text == "No Link")

start.Text = "Start";

//恢复上次的阈值

trackBarMax.Value = Properties.Settings.Default.MaxValue;

trackBarMin.Value = Properties.Settings.Default.MinValue;

labelTrackBarMax.Text = trackBarMax.Value.ToString();

labelTrackBarMin.Text = trackBarMin.Value.ToString();

}

}

/// <summary>

/// 奇偶控制位

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void buttonOddEvenBit\_Click(object sender, EventArgs e)

{

if (buttonOddEvenBit.Tag.ToString() == "true")

{

buttonOddEvenBit.BackColor = SystemColors.Control;

COM.Parity = Parity.None;

buttonOddEvenBit.Tag = "false";

}

else

{

buttonOddEvenBit.BackColor = Color.Lime;

COM.Parity = Parity.Odd;

buttonOddEvenBit.Tag = "true";

}

}

/// <summary>

/// 停止位

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void buttonStopBit\_Click(object sender, EventArgs e)

{

if (buttonStopBit.Tag.ToString() == "true")

{

buttonStopBit.BackColor = SystemColors.Control;

COM.StopBits = StopBits.None;

buttonStopBit.Tag = "false";

}

else

{

buttonStopBit.BackColor = Color.Lime;

COM.StopBits = StopBits.One;

buttonStopBit.Tag = "true";

}

}

/// <summary>

/// 开始按钮

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void start\_Click(object sender, EventArgs e)

{

if (start.Text == "Start")

{

try

{

COM.DataBits = Convert.ToInt32(comboBoxDataBits.SelectedItem.ToString());

COM.BaudRate = Convert.ToInt32(comboBoxBaudRate.SelectedItem);

COM.PortName = comboBoxCom.SelectedItem.ToString();

COM.Open();

}

catch (System.IO.IOException)

{

MessageBox.Show(COM.PortName + "打开失败，请检查连接。");

freshen.Visible = true;

return;

}

catch (System.UnauthorizedAccessException)

{

MessageBox.Show("访问被拒绝，是否还有其他端口监听软件？");

return;

}

start.Text = "Stop";

sendTime();

}

else

{

COM.Close();

start.Text = "Start";

progressBar.Value = 0;

Warring = false;

labelProgressBarValue.Text = "0";

}

}

/// <summary>

/// 协议处理

/// </summary>

char[] trimChar = { 'H', ':' };

/// <summary>

/// 趋势图字段

/// </summary>

Point pointStart, pointValue, pointEnd;

/// <summary>

/// 串口接收

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void COM\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

Control.CheckForIllegalCrossThreadCalls = false;//跨线程操作控件

if (COM.IsOpen)

{

string[] str = new string[5];

try

{

str[0] = COM.ReadLine();

}

catch (System.IO.IOException)

{

textReceived.AppendText("IOerror!\r\n");

}

catch (Exception)

{

MessageBox.Show(e.ToString());

}

if (str[0] == null || str[0].Length == 0)

{

return;

}

if (str[0] == "Boot")

{

COM.WriteLine("Link");

sendTime();

}

if (str[0] == "Time")

{

timerLink.Interval = 2000;

sendTime();

}

if (str[0] == "Alarm")

{

timerLink.Interval = 2000;

COM.WriteLine("");

COM.WriteLine("Aon");

}

if (str[0] == "Lift alarm")

{

timerLink.Interval = 2000;

}

if (str[0][0] == 'H')

{

textReceived.AppendText(str[0] + '\r' + '\n');

str = str[0].Split(',');

str[0] = str[0].TrimStart('H');

try

{

progressBar.Value = int.Parse(str[0]);

labelProgressBarValue.Text = progressBar.Value.ToString();

}

catch (System.FormatException)

{

textReceived.AppendText("Format Error!\r\n");

}

catch (Exception)

{

MessageBox.Show(e.ToString());

}

if (progressBar.Value > trackBarMax.Value || progressBar.Value < trackBarMin.Value)

{

Warring = true;

}

else

{

Warring = false;

}

//以下为panel绘图部分

Pen pen = new Pen(Color.Red, 2);

Pen clear = new Pen(Color.Blue, 3);

Graphics g = panelpicture.CreateGraphics();

pointValue.Y = 100 - progressBar.Value;

pointStart.Y = 100;

g.DrawLine(Pens.White, pointStart, pointEnd);

g.DrawLine(clear, pointStart, pointValue);

g.DrawLine(pen, pointStart, pointValue);

if (pointStart.X < panelpicture.Width)

{

pointStart.X++;

pointValue.X++;

pointEnd.X++;

}

else

{

pointStart.X = 0;

pointValue.X = 0;

pointEnd.X = 0;

}

}

}

}

/// <summary>

/// 窗口关闭

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

this.Controls.Clear();

if (start.Text == "Start")

{

COM.Close();

}

}

/// <summary>

/// 调节阈值

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void trackBar\_Scroll(object sender, EventArgs e)

{

TrackBar T = (TrackBar)sender;

if (T.Value < trackBarMin.Value)

{

trackBarMin.Value = T.Value;

}

if (trackBarMax.Value < T.Value)

{

trackBarMax.Value = T.Value;

}

labelTrackBarMax.Text = trackBarMax.Value.ToString();

labelTrackBarMin.Text = trackBarMin.Value.ToString();

if (start.Text == "Stop")

{

if (progressBar.Value > trackBarMax.Value || progressBar.Value < trackBarMin.Value)

{

Warring = true;

}

else

{

Warring = false;

}

}

Properties.Settings.Default.MaxValue = trackBarMax.Value;

Properties.Settings.Default.MinValue = trackBarMin.Value;

Properties.Settings.Default.Save();

}

/// <summary>

/// 发送文字到下位机

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textSend\_KeyDown(object sender, KeyEventArgs e)

{

if (textSend.Text == null)

{

return;

}

if (e.KeyCode == System.Windows.Forms.Keys.Enter)

{

if (!COM.IsOpen)

{

MessageBox.Show("未连接串口");

return;

}

Bitmap bmp = new Bitmap(320, 24);

Graphics g = Graphics.FromImage(bmp);

g.Clear(Color.White);

g.DrawString(textSend.Text, textSend.Font, Brushes.Black, 0, 0);

COM.WriteLine("\r\n");

COM.WriteLine("Pclear");

COM.WriteLine("\r\n");

for (int y = 0; y < 24; y++)

{

COM.WriteLine("Py" + ((char)y).ToString());

for (int x = 0; x < 320; x++)

{

if (bmp.GetPixel(x, y).GetBrightness() < 0.8f)

COM.WriteLine("P1");

else

COM.WriteLine("P0");

}

}

COM.WriteLine("Pover");

}

}

private bool warring = false;

/// <summary>

/// Warring属性

/// </summary>

public bool Warring

{

get

{

return warring;

}

set

{

if (COM.IsOpen)

{

if (value != warring)

{

if (value == true)

{

COM.WriteLine("Aon");

}

else

{

COM.WriteLine("Aoff");

}

warring = value;

warringChange();

}

if (warring == true)

{

COM.WriteLine("Aon");

}

}

}

}

/// <summary>

/// 报警函数 被属性所调用

/// </summary>

private void warringChange()

{

SoundPlayer player = new SoundPlayer(Properties.Resources.ResourceManager.GetStream("warring"));

if (Warring == true)

{

player.PlayLooping();

}

else

{

player.Stop();

}

}

/// <summary>

/// 悬浮窗控制字段

/// </summary>

private Point mouseOffset;

private Point lastCilck;

/// <summary>

/// 悬浮窗控制

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void progressBar\_MouseDown(object sender, MouseEventArgs e)

{

if (this.Name == "Form2")

{

mouseOffset = new Point(-e.X, -e.Y);

}

if (e.Location == lastCilck)

{

if (this.Name == "Form2")

{

showForm1();

}

else

{

hideForm1();

}

}

else

{

lastCilck = e.Location;

}

}

/// <summary>

/// 悬浮窗控制

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void progressBar\_MouseMove(object sender, MouseEventArgs e)

{

if (this.Name == "Form2" && e.Button == MouseButtons.Left)

{

if (MousePosition != mouseOffset)

{

Point mousePos = Control.MousePosition;

mousePos.Offset(mouseOffset.X, mouseOffset.Y);

this.Location = mousePos;

this.PerformLayout();

}

}

}

/// <summary>

/// 文本框水印

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textSend\_TextChanged(object sender, EventArgs e)

{

if (textSend.TextLength == 0)

{

Labelsend.Visible = true;

}

else

{

Labelsend.Visible = false;

}

}

/// <summary>

/// 文本框水印

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Lablesend\_Click(object sender, EventArgs e)

{

textSend.Focus();

}

/// <summary>

/// 保存接收框内的数据

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void save\_Click(object sender, EventArgs e)

{

Control.CheckForIllegalCrossThreadCalls = false;//跨线程操作控件

FileStream f;

f = File.OpenWrite("rec.txt");

String s = '[' + DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss") + ']';

s += "\r\n\r\n";

s += textReceived.Text;

textReceived.Clear();

f.Position = f.Length;

f.Write(Encoding.Default.GetBytes(s), 0, s.Length);

f.Close();

}

/// <summary>

/// 发送当前时间

/// </summary>

private void sendTime()

{

//空行清除中断计数变量，防止数据出现异常。

COM.WriteLine("");

DateTime time = DateTime.Now;

time = time.AddSeconds(2);//MCU接收加转换的时间，随芯片与算法而定。

COM.WriteLine('T' + time.ToString("yyyyMMdd HHmmss") + '\n');

}

/// <summary>

/// Link信号发生器

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timerLink\_Tick(object sender, EventArgs e)

{

if (COM.IsOpen)

{

COM.WriteLine("Link");

}

}

/// <summary>

/// 时钟校准定时器

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timerTimeCalibration\_Tick(object sender, EventArgs e)

{

if (COM.IsOpen)

{

sendTime();

}

}

/// <summary>

/// 显示悬浮窗

/// </summary>

private void hideForm1()

{

//隐藏所有控件

this.Controls.Clear();

//但是显示progressBar

this.Controls.Add(this.progressBar);

// Form

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(100, 300);

this.Controls.Add(this.progressBar);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;

this.Name = "Form2";

this.ResumeLayout(false);

this.TopMost = true;

//progressBar属性更改

this.progressBar.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.progressBar.Location = new System.Drawing.Point(0, 0);

this.progressBar.Margin = new System.Windows.Forms.Padding(0);

//this.progressBar.Size = new System.Drawing.Size(100, 300);

}

/// <summary>

/// 恢复窗口

/// </summary>

private void showForm1()

{

this.Controls.Clear();

//progressBar恢复

this.progressBar.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Right)));

this.progressBar.Location = new System.Drawing.Point(420, 38);

this.progressBar.Margin = new System.Windows.Forms.Padding(1);

this.progressBar.Size = new System.Drawing.Size(100, 300);

//Form恢复

this.Name = "Form1";

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(584, 461);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Sizable;

this.TopMost = false;

//显示其他组件

this.Controls.Add(this.save);

this.Controls.Add(this.progressBar);

this.Controls.Add(this.panelpicture);

this.Controls.Add(this.labelProgressBarValue);

this.Controls.Add(this.labelTrackBarMin);

this.Controls.Add(this.labelTrackBarMax);

this.Controls.Add(this.freshen);

this.Controls.Add(this.trackBarMin);

this.Controls.Add(this.trackBarMax);

this.Controls.Add(this.textReceived);

this.Controls.Add(label3);

this.Controls.Add(this.buttonStopBit);

this.Controls.Add(this.buttonOddEvenBit);

this.Controls.Add(label2);

this.Controls.Add(this.comboBoxDataBits);

this.Controls.Add(this.comboBoxBaudRate);

this.Controls.Add(this.start);

this.Controls.Add(label1);

this.Controls.Add(this.comboBoxCom);

this.Controls.Add(this.Labelsend);

this.Controls.Add(this.textSend);

this.PerformLayout();

//panel图从头开始绘画

pointStart.X = 0;

pointValue.X = 0;

pointEnd.X = 0;

}

}