

宁宁波财经学院

《单片机原理及应用》

期末大作业

|  |  |
| --- | --- |
| 作业名称 | **《基于STC8A8K64D4的AD电压表及温度计的设计与实现》** |
| 所在学院 | **数字技术与工程学院** |
| 班 级 | **22计科(人工智能)1班** |
| 姓 名 | 王浩然 |
| 学 号 | **2222510131** |

2023年6月20日

1 引言

1.1 课题背景

随着科技的不断进步，人们对测量和监控环境中的各种参数越来越感兴趣。温度和电压是其中两个最基本的参数，对于许多电子设备和系统的正常运行至关重要。因此，设计和实现一个能够准确测量和监控温度和电压的仪表系统具有重要的实际意义。

本课题旨在设计和实现基于STC8A8K64D4的AD电压表和温度计，具有较为准确测量和显示温度和电压；串口通信：能够将测量数据上传到上位机；温度报警：当测量数据超过预设的门限值时，能够发出警报等功能。

为了实现上述功能，本课题需要进行以下工作：选择合适的热敏电阻传感器，并进行电路设计和调试；利用现有的单片机，进行程序编写和调试；设计串口通信协议，并进行串口通信程序的编写和调试；设计报警电路，并进行报警程序的编写和调试。

1.2国内外研究现状

目前，国内外对于温度和电压测量的研究已经非常广泛和深入。在温度测量方面，许多研究者提出了不同的方法，如热电偶、热电阻、集成温度传感器等。在电压测量方面，则主要依赖于AD转换器来实现。同时，许多研究者致力于提高仪表系统的精度和稳定性，以及研究新的测量方法和传感器技术，以满足不同领域的需求。

在国内方面，目前也有许多研究者关注于基于STC8A8K64D4的AD电压表和温度计的研究和应用。例如，有研者基于STC8A8K64D4设计了一种高精度数字温度计和数字电压表，采用热敏电阻作为温度传感器，AD5933作为电压基准源，实现了高精度的温度和电压测量。还有研者基于STC8A8K64D4设计了智能家居环境监测系统，实现了温湿度、气压、光照等环境参数的测量和上传至云端服务器。这些研究为本课题的研究提供了重要的参考和借鉴。

2 算例分析

2.1 电压测算温度分析

使用的计算公式如下：

(5.1a)

(5.1b)

其中，为温度值， 为参考温度值(这里取 )， 为 NTC 材料参数， 为 NTC 电阻值，​ 为 下的 NTC 电阻值。

图表, 折线图

描述已自动生成通过模型计算得出的结果见表1。表中行是由电压测算阻值的结果，列是阻值对应的温度。

表1阻值对应温度表

1 引言

1.1概要

本章介绍了基于STC8A8K64D4的AD电压表和温度计的设计与实现项目的选题背景、应用场景和国内外研究现状。该项目的应用价值在于可以广泛应用于各种电子设备和系统的温度和电压测量，例如电力系统、家用电器、工业自动化等领域，还可以为环境监测、工业过程控制等领域提供数据支持。在研究方面，该项目的精度和稳定性还有待进一步提高，同时可以探索更多的应用场景，例如与其他物联网技术的结合、智能化数据分析等。基于STC8A8K64D4的AD电压表和温度计的设计与实现具有很高的应用价值和社会背景，相信在不断深入研究和探索的过程中，将会有更加广泛的应用前景。

1.2选题背景

当前社会，随着电子技术和物联网技术的快速发展，温度和电压测量已成为各行各业的基本需求。尤其在电力系统、工业自动化、智能家居等领域，温度和电压测量更是不可或缺的一部分。随着人们生活水平的提高，对于温度和电压测量的精度和稳定性的要求也越来越高。

本课题的研究成果可以应用于各种电子设备和系统，例如电力系统、家用电器、工业自动化等。在电力系统方面，该系统可以应用于电力设备的运行监测和故障诊断，通过测量设备的温度和电压，及时发现设备故障隐患，保障电力系统的安全稳定运行。在家用电器方面，该系统可以应用于智能家居系统，实现设备的远程监控和控制。在工业自动化方面，该仪表系统可以应用于各种工业生产设备中，实现设备的温度和电压监测，提高设备的可靠性和生产效率。

本课题的研究还可以为环境监测、工业过程控制等领域提供数据支持，具有重要的应用价值。在环境监测方面，该成果可以应用于多种环境参数的测量，例如温度等，为环境保护和监测提供数据支持。在工业过程控制方面，该成果可以应用于各种工业过程的控制和监测，提高生产效率和产品质量。

2  **总体设计**

2.1对系统的功能做出详细的描述，需求分析：

本系统是一个基于STC8A8K64D4的AD电压表及温度计系统，它通过使用内置12位AD和热敏电阻来测量温度和电压。系统的主要功能包括温度和电压的测量、显示和串口输出，以及温度报警和LED灯指示功能。下面是对系统功能的详细描述和需求分析。

2.1.1温度测量和显示

系统使用热敏电阻来测量温度，通过内置12位AD进行采样和转换，将温度值转换为数字信号进行处理和显示。系统可以在数码管和串口之间同时显示温度值，温度单位为摄氏度。

需求分析：系统需要具备高精度的温度测量能力，能够准确反映被测环境的温度变化。同时，系统应该能够以不同的方式显示温度值。

2.1.2电压测量和显示

系统通过内置12位AD进行电压采样和转换，将电压值转换为数字信号进行处理和显示。系统可以在数码管和串口之间同时显示电压值，电压单位为伏特。

需求分析：系统需要具备高精度的电压测量能力，能够准确反映被测电压的变化。同时，系统应该能够以不同的方式显示电压值。

2.1.3温度报警功能

当温度传感器检测到温度大于等于30℃时，系统会通过蜂鸣器发出2秒钟的鸣叫声来进行报警。

需求分析：系统需要具备温度报警功能，以确保被测环境温度不超过安全范围。同时，系统应该能够提供灵活的温度报警阈值和报警方式。

2.1.4 LED灯指示功能

当温度从20℃开始每升高2℃，系统会多点亮一盏LED灯。

需求分析：系统需要具备LED灯指示功能，以直观地反映被测环境温度的变化情况。同时，系统应该能够提供灵活的LED灯指示方式和指示位置。

2.1.5按键控制

系统使用按键来实现温度和电压之间的切换，以方便用户在不同的使用场景下查看温度或电压值。

需求分析：系统需要具备按键控制功能，以便用户能够灵活地切换温度和电压之间的显示。同时，系统应该能够提供稳定可靠的按键控制方式，以确保用户的操作体验和使用效果。

2.1.6 串口通信

系统可以通过串口将测量结果发送到计算机，在串口助手中查看详细的电压或温度变化。

需求分析：系统通过串口将数据发送至PC机，便于用户对数据的保存、统计与分析，直观显示了温度(电压)随时间、环境等变化，对系统提供了扩展空间。

2.1.7低功耗设计

本系统的应用场景可能比较广泛，因此需要考虑低功耗设计，以满足长时间使用的需求。

需求分析：系统需要具备低功耗设计，以确保在各种应用场景下都能够稳定运行并延长电池寿命。同时，系统应该能够提供灵活的功耗控制方式。

综上所述，本基于STC8A8K64D4的AD电压表及温度计系统需要具备温度和电压的测量、显示和串口输出功能、温度报警功能、LED灯指示功能、按键控制功能以及低功耗设计等特点，以满足不同用户的需求。

2.2本设计搭建的系统环境，安装和使用到的软件，以及计算机硬件配置：

**(1)**本设计系统环境的搭建需要满足设计的开发和运行所需的性能和功能要求。

在本次系统中，我选择了Windows 11 专业版作为操作系统，版本号为22H2，安装日期为‎2023/‎3/‎14，操作系统版本为22631.1835，体验为Windows Feature Experience Pack 1000.22644.1000.0。该操作系统具有更好的用户体验，更高的性能和更强大的安全性能，可以满足我们的需求。

**(2)**为了进行系统设计和开发，我安装使用了以下软件：

GIT：用于对项目进行版本控制和将项目开源，以便可以更好地管理代码，并向其他人共享代码。

STC-ISP V6.91N：用于将代码烧录入单片机中以及通过串口助手从单片机中接受数据并显示。

Keil：用于进行调试和烧录代码，支持多种单片机芯片，可以方便地进行编程和调试。

Microsoft VS Code Insiders：用于编写代码和代码管理，具有多种编程语言的支持，并提供了许多有用的插件和工具。

MinGW：用于编译项目，并提供所需头文件、库等，以便我们可以方便地进行编译和构建。

Notepad++：用于查看、修改和编写代码，是一款轻量级的文本编辑器，支持多种编程语言。

TortoiseGit：用于对项目进行版本控制，具有简单易用的界面和丰富的功能。

Typora：用于编写说明文档和注意事项，具有丰富的排版和格式化功能，可以方便地生成漂亮的文档。

**(3)**在硬件配置方面，我们需要一台性能强劲的计算机，以支持系统的开发和测试。我选择了以下硬件配置：

处理器：AMD Ryzen 7 7735H with Radeon Graphics，具有八个核心和超线程技术，可以提供更快的处理速度和更好的运行效率。

主板：MECHREVO WUJIE 14，具有快速传输数据的高速总线和丰富的扩展接口。

内存：镁光 16GB LPDDR5 6400MHz (4GB x 4)，具有更快的读写速度和更高的容量，可以方便地存储和处理大量数据。

显卡：AMD Radeon(TM) Graphics (512MB)，具有良好的性能和流畅的图像处理能力。

显示器：NE140QDM-NX2京东方 BOEOAF2，具有高分辨率和良好的显示效果，可以方便地进行开发和测试。

磁盘：INTEL SSDPEKNU512GZ (512GB)，具有更快的读写速度和更大的容量，可以存储和处理大量数据。

声卡：Senary Audio、AMD BT Audio Device、AMD High Definition Audio Device，可以提供良好的音频效果和体验。

网卡：RZ608 Wi-Fi 6E 80MHz，可以提供更快的网络速度和更稳定的连接。

2.3 总体框图、硬件描述

图示

描述已自动生成2.3.1总体设计

图1总体框图

**1**初始化系统和外设，包括ADC模块、数码管和蜂鸣器等。

**2**进入一个无限循环，每次循环都会读取ADC通道的电压值，并将其转换为温度值。

**3**根据温度值设置数码管的显示内容，以及控制蜂鸣器的鸣叫状态。

**4**如果温度低于28℃，则设置蜂鸣器鸣叫状态为1。

**5**如果温度高于30℃且蜂鸣器鸣叫状态为1，则停止蜂鸣器鸣叫。

**6**循环回到第2步，继续读取ADC通道的电压值并更新温度值。

2.3.2硬件描述

硬件设备在项目中发挥着重要作用。在该系统中，使用了以下硬件设备：

**(1)**STC8A8K64D开发板

STC8A8K64D开发板是该项目的核心板，具有高性能、低功耗、易于编程、易于扩展等特点。开发板通过与其他硬件设备的连接，实现了该系统的各种功能，包括电压测量、温度测量、数码管显示、串口通信、温度报警、LED灯显示等。

**(2)**MicroUSB-TypeC数据线

MicroUSB-TypeC数据线用于将开发板连接到PC主机上，以便进行程序调试和数据传输。该数据线采用了通用的TypeC接口，具有快速传输速度和稳定性，可以有效提高系统的工作效率。

**(3)**500Ω电阻10个

500Ω电阻用于对电路中的电流进行限制和调节。在该项目中，该电阻主要用于串联提高上拉电阻进行电压调节，以保证数据显示的准确性和稳定性。

**(4)**104电容一个

104电容用于对电路中的电压进行平滑处理和滤波。在该系统中，该电容主要用于对AD转换器进行滤波，以保证电压测量的准确性和稳定性。

**(5)**MF52热敏电阻一个

MF52热敏电阻用于测量环境温度。在该系统中，该热敏电阻通过与AD转换器连接，可以将环境温度转化为电压信号，再进行数字化处理，以实现温度的测量和显示。

**(6)**四位数码管一个

四位数码管用于显示电压和温度数值。在该系统中，该数码管通过与开发板的IO口连接，可以将数字信号转化为数码显示，以实现电压和温度的直观显示。

**(7)**杜邦线若干条

杜邦线用于连接各种硬件设备，如开发板、电阻、电容、热敏电阻、数码管等。在该系统中，杜邦线起到了连接各种硬件设备的作用，以实现各项功能。

**(8)**PC主机一台

PC主机用于程序编写、调试和数据处理。在该系统中，PC主机通过与开发板的连接，可以实现对系统的程序编写、调试和数据传输，以满足系统的各项需求。

3  **详细设计**

3.1主控模块的设计与实现

3.1.1主控模块主要实现以下功能：

1）通过AD采集温度和电压信号，进行测量并显示在数码管上；

2）通过串口通信，将采集数据发送给上位机进行处理；

3）当温度超过设定值时，通过LED灯和蜂鸣器进行报警提示。

3.1.2模块硬件设计

本模块硬件设计主要包括以下模块：

1）STC8A8K64D4单片机模块：作为系统的主控芯片，实现数据采集、处理和控制等功能；

2）AD模块：用于采集温度和电压信号；

3）数码管模块：用于显示温度和电压值；

4）LED模块：用于进行报警提示；

5）蜂鸣器模块：用于进行报警提示。

3.1.3模块软件设计

本模块软件设计主要包括以下模块：

1）初始化模块：对单片机的各个外设进行初始化；

2）AD采集模块：对温度和电压进行采集，并进行相应的数据处理；

3）数码管显示模块：将处理后的数据显示在数码管上；

4）串口通信模块：将采集的数据通过串口发送给上位机；

5）温度报警模块：当温度超过设定值时，进行报警提示。

3.2 AD转换模块的设计与实现

图示, 表格

中度可信度描述已自动生成STC8A8K64D4系列单片机内部集成了一个12位高速A/D转换器。ADC的时钟频率为系统频率2 分频再经过用户设置的分频系数进行再次分频（ADC的工作时钟频率范围为SYSclk/2/1到SYSclk/2/16）。

ADC\_POWER：ADC电源控制位 0：关闭ADC电源 1：打开ADC电源。  
ADC\_START：ADC转换启动控制位。写入1后开始ADC转换，转换完成后硬件自动将此位清零。 0：无影响。即使ADC已经开始转换工作，写0也不会停止A/D转换。 1：开始ADC转换，转换完成后硬件自动将此位清零。 ADC\_FLAG：ADC转换结束标志位。当ADC完成一次转换后，硬件会自动将此位置1，并向CPU提出 中断请求。此标志位必须软件清零。

ADC\_EPWMT：使能PWM实时触发ADC功能。

ADC\_CHS[3:0]：ADC模拟通道选择位

ADC\_CHS[3:0] ADC通道

0000 P1.0/ADC0

0001 P1.1/ADC1

0010 P1.2/ADC2

0011 P1.3/ADC3

0100 P1.4/ADC4

0101 P1.5/ADC5

0110 P1.6/ADC6

0111 P1.7/ADC7

1000 P0.0/ADC8

1001 P0.1/ADC9

1010 P0.2/ADC10

1011 P0.3/ADC11

1100 P0.4/ADC12

1101 P0.5/ADC13

1110 P0.6/ADC14

1111 测试内部1.19V

3.2.1 AD模块设计与实现

AD模块是本设计中一个重要的模块，其主要功能是将模拟信号转换为数字信号。在本设计中，使用单片机内置的ADC模块实现AD模块。

3.2.1 AD模块功能概述

AD模块的主要功能是采集外部的模拟信号，并将其转换为数字信号，以便于单片机进行处理和分析。在本设计中，AD模块的输入信号为温度传感器输出的模拟信号，输出为对应的数字信号。

3.2.2 AD模块硬件设计与实现

AD模块的硬件设计与实现主要包括以下几个方面：

1. 选择ADC模块：本设计中选择了单片机内置的ADC模块，具有较高的精度和采样率。

2. 选择采样通道：在本设计中，选择了单片机的P0.0口作为ADC的采样通道。

3. 设计电路：为了保证ADC模块的正常工作，需要设计合适的电路，包括电源电压调节、滤波电路等。

4. 程序设计：为了实现AD模块的功能，需要在程序中设置ADC的相关参数，并编写相应的读取ADC数值的程序。

3.2.3 AD模块软件设计与实现

AD模块的软件设计与实现主要包括以下几个方面：

1. 设置ADC参数：在程序中设置ADC的采样通道、分辨率、采样率等参数。

2. 启动ADC转换：在程序中启动ADC转换，并等待转换完成。

3. 读取ADC数值：在转换完成后，读取ADC转换的数值，并进行相应的处理，例如根据公式计算温度。

4. 程序优化：为了提高程序的运行效率和准确性，需要进行相应的程序优化。

3.2.4 AD模块测试结果

经过测试，本设计中的AD模块能够正常采集外部模拟信号，并将其转换为数字信号。在本设计中，采用了NTC热敏电阻作为温度传感器，实现了温度的采集和转换。测试结果表明，AD模块的精度和稳定性较高，能够满足本设计的需求。

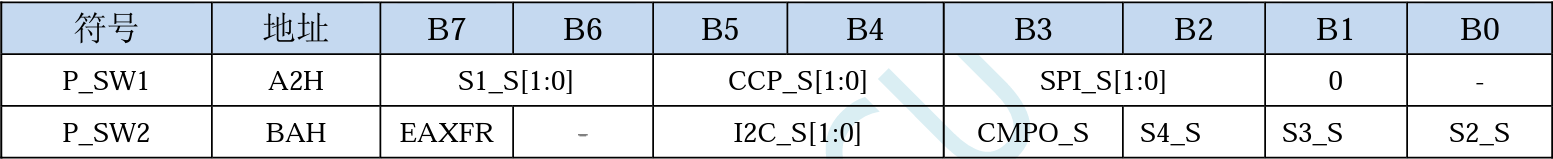
3.3 串口通信模块

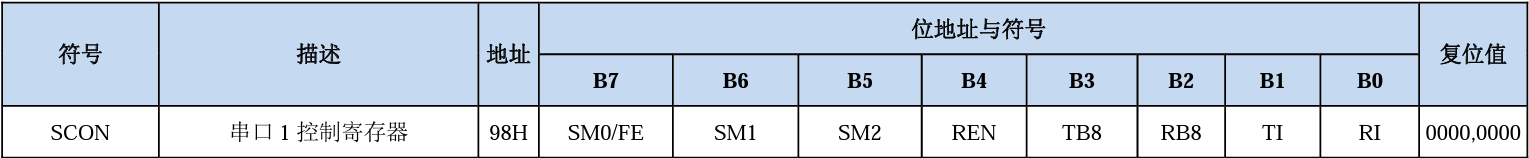
3.3.1串口通信介绍

STC8A8K64D4系列单片机具有4个全双工异步串行通信接口。每个串行口由2个数据缓冲器、一 个移位寄存器、一个串行控制寄存器和一个波特率发生器等组成。每个串行口的数据缓冲器由2个互相 独立的接收、发送缓冲器构成，可以同时发送和接收数据。

STC8A8K64D4系列单片机的串口1有4种工作方式，其中两种方式的波特率是可变的，另两种是固定的，以供不同应用场合选用。串口2/串口3/串口4都只有两种工作方式，这两种方式的波特率都是可变的。用户可用软件设置不同的波特率和选择不同的工作方式。主机可通过查询或中断方式对接收/发送进行程序处理，使用十分灵活。 串口1、串口2、串口3、串口4的通讯口均可以通过功能管脚的切换功能切换到多组端口，从而可 以将一个通讯口分时复用为多个通讯口。

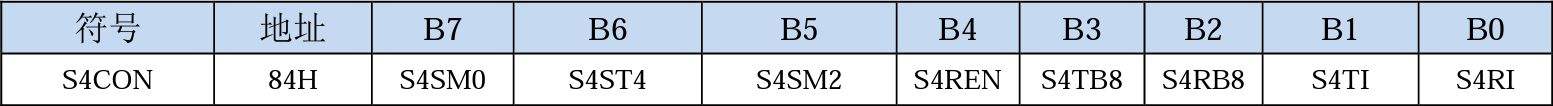
3.3.2 串口功能脚切换



表格

描述已自动生成3.3.3 串口寄存器

在本设计中使用串口4，故以串口4做说明



S4SM0：指定串口4的通信工作模式，如下表所示：

表格

描述已自动生成

S4ST4：选择串口4的波特率发生器 0：选择定时器2为串口4的波特率发生器 1：选择定时器4为串口4的波特率发生器

S4SM2：允许串口4在模式1时允许多机通信控制位。在模式1时，如果S4SM2位为1且S4REN位为 1，则接收机处于地址帧筛选状态。此时可以利用接收到的第9位（即S4RB8）来筛选地址帧： 若S4RB8=1，说明该帧是地址帧，地址信息可以进入S4BUF，并使S4RI为1，进而在中断服务 程序中再进行地址号比较；若S4RB8=0，说明该帧不是地址帧，应丢掉且保持S4RI=0。在模式1 中，如果S4SM2位为0且S4REN位为1，接收收机处于地址帧筛选被禁止状态。不论收到的S4RB8 为0或1，均可使接收到的信息进入S4BUF，并使S4RI=1，此时S4RB8通常为校验位。模式0 为非多机通信方式，在这种方式时，要设置S4SM2 应为0。

S4REN：允许/禁止串口接收控制位 0：禁止串口接收数据 1：允许串口接收数据

S4TB8：当串口4使用模式1时，S4TB8为要发送的第9位数据，一般用作校验位或者地址帧/数据帧标 志位，按需要由软件置位或清0。在模式0中，该位不用。

S4RB8：当串口4使用模式1时，S4RB8为接收到的第9位数据，一般用作校验位或者地址帧/数据帧标 志位。在模式0中，该位不用。

S4TI：串口4发送中断请求标志位。在停止位开始发送时由硬件自动将S4TI置1，向CPU发请求中断， 响应中断后S4TI必须用软件清零。

S4RI：串口4接收中断请求标志位。串行接收到停止位的中间时刻由硬件自动将S4RI置1，向CPU发 中断申请，响应中断后S4RI必须由软件清零。

图示, 表格

中度可信度描述已自动生成

4  **测试与调试**

4.1 调试

在完成大作业的过程中，调试是一个非常重要的环节。在调试过程中，可能会遇到各种各样的问题，如何解决这些问题直接关系到大作业的完成质量和进度。本章主要针对在完成大作业过程中遇到的问题进行了详细的描述，并给出了相应的解决方案。

问题一：数据溢出

在将AD值转换为电压时，由于编译器无法自动扩大数据范围，导致出现了数据溢出的现象。针对这个问题，我使用了串口输出数据不断修改的方法，最终更改了数据类型解决了问题。

解决方案：

更改数据类型：将原来的数据类型更改为更大的数据类型，例如将short改为int或者long，这样可以扩大数据范围，避免数据溢出的问题。

问题二：REF引脚未接入基准电压

单片机上的REF引脚在一开始未接入基准电压，造成测得数据不正确。针对这个问题，我们将REF引脚接入基准电压后便解决了问题。

解决方案：

将REF引脚接入基准电压：REF引脚是单片机的参考电压引脚，它用于提供AD转换所需的参考电压。因此，如果REF引脚未接入基准电压或者接入的基准电压不正确，就会导致测得的数据不正确。将REF引脚正确接入基准电压后，就可以得到正确的测量结果。

4.2总结与展望

这是一个综合性的项目，涉及到多个方面的知识和技能。在未来的研究中，可以从以下几个方面进行优化和探索：

1)提高系统的测量精度和稳定性：目前系统中使用了一些技巧和方法来提高测量精度和稳定性，例如多次采样和平均值滤波等。然而，这些方法仍然存在一些局限性，例如噪声干扰和信号漂移等。未来可以探索更多的滤波算法和技术，例如卡尔曼滤波、神经网络滤波等，以提高系统的测量精度和稳定性。

2)加强系统的可靠性和安全性：在应用中，系统的可靠性和安全性至关重要。未来可以加强系统的安全性和可靠性，例如增加数据校验位、加密通讯、防电磁干扰等措施。

3)探索更多的应用场景：可以应用于许多领域，例如工业自动化、智能家居、医疗保健等。未来可以探索更多的应用场景，例如与其他物联网技术的结合、智能化数据分析，可以将该系统与云计算、大数据等技术相结合，为用户提供更加智能化、个性化的服务等。

未来，随着技术的不断发展和进步，该项目将会越来越完善和普及，为人们的生活和工作带来更多的便利和效益。

附 录

附录1 主要功能模块代码

*/\*-------------------------------------------------------------------------------*

*/\*       (C) Copyright 2023, PSYM. All Rights Reserved*

*/\*-------------------------------------------------------------------------------*

*/\* File name   : AD.c*

*/\* Description :*

*/\* Author      : PSYM*

*/\*-------------------------------------------------------------------------------*

*/\* Update note:*

*/\* -----------   -----------   --------------------------------------------------*

*/\*    Date         Author                              Note*

*/\* -----------   -----------   --------------------------------------------------*

*/\* 2023-06-08       PSYM*

*/\*   14:50*

*\*\*/*

*/\* ---------------------------------- 包含头文件 ---------------------------------- \*/*

*#include "AD.h"*

*/\* ---------------------------------- 私有宏定义 ---------------------------------- \*/*

*#define VCC 3300 // 电源电压mv*

*#define ADC\_RES\_12BIT 4096 // 12位采样分辨率*

*/\* ---------------------------------- 私有类型定义 ---------------------------------- \*/*

*sbit ADC\_IO = P0 ^ 0; // ADC通道选择端口*

*sbit ADC\_CFG = 0xDE; // ADC配置寄存器*

*/\* ---------------------------------- 私有变量 ---------------------------------- \*/*

*uint16 ADC\_Value = 0; // ADC转换结果*

*int16 tmp = 0;      // 转换温度值*

*uint32 voltage = 0; // 实际电压值*

*/\* ---------------------------------- 扩展变量 ---------------------------------- \*/*

*/\* None. \*/*

*/\* ---------------------------------- 私有函数原型 ---------------------------------- \*/*

*void AD\_Start();*

*/\* ---------------------------------- 函数体 ---------------------------------- \*/*

*/\*\**

*\* @brief: None*

*\* @param: None*

*\* @return: None*

*\* @note: None*

*\*/*

*void AD\_Init()*

*{*

*ADC\_CONTR &= 0xF0; // 清除模拟通道选择位*

*ADC\_CONTR |= 0x08; // 设置ADC模拟通道 P0.0*

*ADC\_CFG &= 0x00; // 清除所有标志位*

*ADC\_CFG |= 0x00; // 设置ADC转换结果为左对齐*

*ADC\_CFG |= 0x0F; // 设置ADC时钟频率*

*AD\_Start(); // 开启ADC转换*

*}*

*void AD\_Start()*

*{*

*ADC\_CONTR |= 0x80; // 开启ADC电源*

*Delay\_ms(1);*

*ADC\_CONTR |= 0x40; // 启动ADC转换*

*}*

*/\*\**

*\* @brief: 采样转换*

*\*/*

*void Sampl()*

*{*

*AD\_Start();*

*while (!(ADC\_CONTR & 0x20))*

*; //等待ADC转换完成*

*ADC\_Value = ADC\_RES << 4;*

*ADC\_Value |= ADC\_RESL >> 4;*

*voltage = VCC \* (uint32)ADC\_Value / ADC\_RES\_12BIT; // 电压值*

*tmp = NTC\_TempValue\_Calculate(voltage); // 通过公式求得温度、*

*ADC\_CONTR &= 0xDF; //清除ADC转换完成标志位*

*}*

*/\*>>>>>>>>>> (C) COPYRIGHT PSYM <<<<<< >>>>>> END OF FILE <<<<<<<<<<\*/*

*/\*-------------------------------------------------------------------------------*

*/\*       (C) Copyright 2023, PSYM. All Rights Reserved*

*/\*-------------------------------------------------------------------------------*

*/\* File name   : ntc\_temp\_calc.c*

*/\* Description :*

*/\* Author      : PSYM*

*/\*-------------------------------------------------------------------------------*

*/\* Update note:*

*/\* -----------   -----------   --------------------------------------------------*

*/\*    Date         Author                              Note*

*/\* -----------   -----------   --------------------------------------------------*

*/\* 2023-06-14       PSYM*

*/\*   09:16*

*\*\*/*

*/\* ---------------------------------- 包含头文件 ---------------------------------- \*/*

#include "ntc\_temp\_calc.h"

*/\* ---------------------------------- 私有宏定义 ---------------------------------- \*/*

#define T25 (25 + 273.15) *// T25 = 25 + 273.15*

#define R25 (10) *// 25摄氏度下对应的NTC电阻值为10千欧，注意此处单位为千欧，因此传入的Rntc单位也要一致为千欧*

#define RES\_UP (5.0) *// 上拉采样电阻 RES\_UP = 5.0K*

#define AVDD\_MV (3300) *// AD电源电压 3300mv*

#define BX (3950) *// NTC 材料参数 B 值*

#define ABS\_0 (273.15) *//*

*/\* ---------------------------------- 私有类型定义 ---------------------------------- \*/*

*/\* None. \*/*

*/\* ---------------------------------- 私有变量 ---------------------------------- \*/*

*/\* None. \*/*

*/\* ---------------------------------- 扩展变量 ---------------------------------- \*/*

*/\* None. \*/*

*/\* ---------------------------------- 私有函数原型 ---------------------------------- \*/*

*/\* None. \*/*

*/\* ---------------------------------- 函数体 ---------------------------------- \*/*

*/\*\**

*\* @brief: 温度计算函数*

*\* @param: vdd\_value为内部电压值*

*\*       TempADCValue为对应AD检测的电压值（在电路中采样电阻 R 的电压）*

*\* @return: 返回温度值*

*\* @note: 1.需要添加头文件"math.h"*

*\*        2.数学中ln是以e为底数(lne2=2)(ln为自然对数*

*\*         e=2.7182818284590452353602874713527…)*

*\*         math.h只有两个函数log和log10,其中log代表数学上的ln*

*\*         函数 log10(x) 以10为底的对数,即 lg(x),以其它数为底的对数用换底公式来表示*

*\*         此NTC温度计算公式只需要用到log这个函数即可。*

*\*/*

int16 NTC\_TempValue\_Calculate(uint32 *TempADCValue\_mv*)

{

    float tmp = 0;

*/\**

*必须要乘以1.0，才能得到浮点数0.51，否则得到的数值为0；*

*因为：整型/整型=整型，而无法得到浮点型，即 51/100 = 0 ; 而51\*1.0/100 = 0.51*

*\*/*

    tmp = (RES\_UP \* 1.0) / R25; *// R/R0*

    tmp = tmp \* *TempADCValue\_mv* / (AVDD\_MV - *TempADCValue\_mv*); *//\* V/(VDD-V)*

    tmp = log(tmp);

    tmp /= BX; *// /B*

    tmp \*= T25; *// \*(T0+273.15)*

    tmp = T25 / (tmp + 1); *// T*

    tmp -= ABS\_0; *// -273.15*

    return tmp + 0.5;

}

*/\*>>>>>>>>>> (C) COPYRIGHT PSYM <<<<<< >>>>>> END OF FILE <<<<<<<<<<\*/*

*/\*-------------------------------------------------------------------*

*/\*      (C) Copyright 2021-2022, nbufe. All Rights Reserved*

*/\*-------------------------------------------------------------------*

*/\* File name   : DLED.c*

*/\* Description :*

*/\* Author      : nbufe*

*/\*-------------------------------------------------------------------*

*/\* Update note:*

*/\* ---------   -----------   ----------------------------------------*

*/\*   Date        Author                       Note*

*/\* ---------   -----------   ----------------------------------------*

*/\**

*/\**

*\*\*/*

*/\* 包含头文件 -------------------------------------------------------\*/*

#include "DigLED.h"

#include "STC8Ax\_Delay.h"

*/\* 私有类型定义 -----------------------------------------------------\*/*

*/\* None. \*/*

*/\* 私有宏定义 -------------------------------------------------------\*/*

code const uint8\_t DigDisplay\_table[] = *// 标准字库*

{

*//   0     1     2     3     4    5     6     7     8     9     o     C*

    0xFC, 0x60, 0xDA, 0xF2, 0x66,0xB6, 0xBE, 0xE0, 0xFE, 0xF6, 0xC6, 0x9C

};

#define DigShow\_PORT    P1

#define DigShow\_COM     P2

*/\* 私有变量 ---------------------------------------------------------\*/*

unsigned char DisplayBuf[4] = { 0 };

unsigned char Display\_index = 0;*//动态扫描数码管的步骤变量*

*/\* 扩展变量 ---------------------------------------------------------\*/*

*/\* None. \*/*

*/\* 私有函数原形 -----------------------------------------------------\*/*

*/\* None. \*/*

*/\* 函数体 -----------------------------------------------------------\*/*

*/\**

*\* 功能: 驱动数码管的动态扫描程序*

*\* 参数: None*

*\* 返回: None*

*\* 说明:*

*\*/*

void DigLED\_Display()

{

    DigShow\_COM = 0xFF;

    DigShow\_PORT = DisplayBuf[Display\_index];

    DigShow\_COM = \_crol\_(0xFE, Display\_index);

    if (++Display\_index > 3)

        Display\_index = 0; *//1位结束*

}

*/\**

*\* 功能: 在指定位显示数值*

*\* 参数: index 位序*

*\*       number 显示值*

*\*       dot 小数点 1：显示  0：不显示*

*\* 返回: None*

*\* 说明:*

*\*/*

void DisplayLoad(unsigned char *index*, unsigned char *number*, unsigned char *dot*)

{

    DisplayBuf[*index*] = DigDisplay\_table[*number*] | ( *dot* ? 0x01 : 0 ) ; *//小数点为第7位*

}

*/\*>>>>>>>>>> (C) COPYRIGHT NBUFE <<<<<< >>>>>> END OF FILE <<<<<<<<<<\*/*