**目录**

[1 设计题目 1](#_Toc43377060)

[2 设计目的 1](#_Toc43377061)

[3 设计内容及要求 1](#_Toc43377062)

[4 系统总体结构 1](#_Toc43377063)

[5 硬件设计 2](#_Toc43377064)

[5.1 元件选取 2](#_Toc43377065)

[5.2 电路设计 7](#_Toc43377066)

[6 软件设计 8](#_Toc43377067)

[6.1 中断流程图 8](#_Toc43377068)

[6.2 主函数流程图 10](#_Toc43377069)

[6.3 显示函数流程图 11](#_Toc43377070)

[7 硬件调试 12](#_Toc43377071)

[7.1 转速测试 12](#_Toc43377072)

[7.2 数据记录 14](#_Toc43377073)

[8 设计小结 15](#_Toc43377074)

[9 参考文献 15](#_Toc43377075)

[附件1源程序代码 16](#_Toc43377076)

**图目录**

[图 4‑1系统框图 1](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377077)

[图 5‑1 MSP430G2x53微控制器结构 2](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377078)

[图 5‑2 Timer\_功能框图 3](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377079)

[图 5‑3 ST188实物图 4](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377080)

[图 5‑4 ST188外形图 4](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377081)

[图 5‑5 ST188极限参数 4](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377082)

[图 5‑6 ST188电气特性 5](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377083)

[图 5‑7 LM393内部透视图 5](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377084)

[图 5‑8 LM393实物图 5](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377085)

[图 5‑9 LCD控制流程 6](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377086)

[图 5‑10 LCD驱动模块 6](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377087)

[图 5‑11 基于I2C的IO扩展 6](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377088)

[图 5‑12 光电传感器模块 7](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377089)

[图 6‑1 TIMER0中断服务函数 8](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377090)

[图 6‑2 TIMER1中断服务函数 9](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377091)

[图 6‑3 主函数流程图 10](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377092)

[图 6‑4 显示函数流程图 11](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377093)

[图 7‑1 系统上电 12](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377094)

[图 7‑2 转速测试1 12](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377095)

[图 7‑3 转速测试2 13](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377096)

[图 7‑5 转速测试3 13](file:///D:\传感器课程设计\非接触式转速测量表\报告\非接触式转速测量表.docx#_Toc43377097)

**表目录**

[表格 7‑1 调试数据记录 14](#_Toc43377098)

**摘要**

随着现代科技的不断进步，各个领域对测速系统的应用越来越多，同时也对测速精度的要求越来越高。从测速仪器是否与转轴接触又可分为接触式，非接触式，目前常用的是非接触测量，采用传感器为检测元件，读取到的信号通过微处理器来进行计算显示。

目前，常见的单片机有51系列和MSP430系列等。51系列应用广泛、功能完备，但抗干扰能力不强；与上述相比，MSP430系列具有功能强大、集成度高、抗干扰能力强、成本低、市场流通性大、能进入低功耗模式运行，保证产品的时间长、损耗低、精度高、稳定性佳等特点。

综上所述，本文将基于MSP430G2553单片机进行非接触式转速测量仪的设计。

**关键字**：转速测量；单片机；MSP430；光电传感器

**Abstract**

With the continuous progress of modern science and technology, the application of speed measurement system in various fields is increasing, and the requirement of speed measurement accuracy is also becoming higher and higher.

Whether the speed measuring instrument is in contact with the rotating shaft can be divided into contact type and non-contact type. Currently, non-contact measurement is commonly used. The sensor is used as the detection element, and the signal read is calculated and displayed by the microprocessor.

At present, the common SCM has 51 series and MSP430 series.Series 51 is widely used and has complete functions, but its anti-interference ability is not strong.Compared with the above, MSP430 series has powerful functions, high integration, strong anti-interference ability, low cost, large market liquidity, can enter the low-power mode of operation, to ensure the product's long time, low loss, high precision, good stability and other characteristics.

To sum up, this paper will be based on MSP430G2553 microcontroller for the design of non-contact speed measuring instrument.

**Keywords**: speed measurement; single chip microcomputer; MSP430; photoelectric sensor

**非接触式转速测量表**

# 设计题目

非接触式转速测量表。

# 设计目的

运用单片机原理及其应用等课程知识,根据题目要求进行软硬件系统的设计和调试，从而加深对传感器的理解，把学过的比较零碎的知识系统化，比较系统地学习开发单片机应用系统的基本步骤和基本方法，使应用知识能力、设计能力、调试能力以及报告撰写能力等有一定的提高。

# 设计内容及要求

设计一个用光电反射式传感器检测的非接触式转速测量表，检测范围60-3000转/分，精度±5％，显示分辨率0.1转/分。

# 系统总体结构

此次非接触式转速测量表包括：光电反射式传感器，显示模块，控制模块。

系统上电，光电反射式传感器的电源由MSP430G2553开发板上的3.3V电源提供。单片机的1个IO口捕获光电反射式传感器输出的方波周期，最后借助软件编程将转速显示在128段液晶显示屏上。系统总设计框图如图 4‑1下：



图 4‑1系统框图

# 硬件设计

## 元件选取

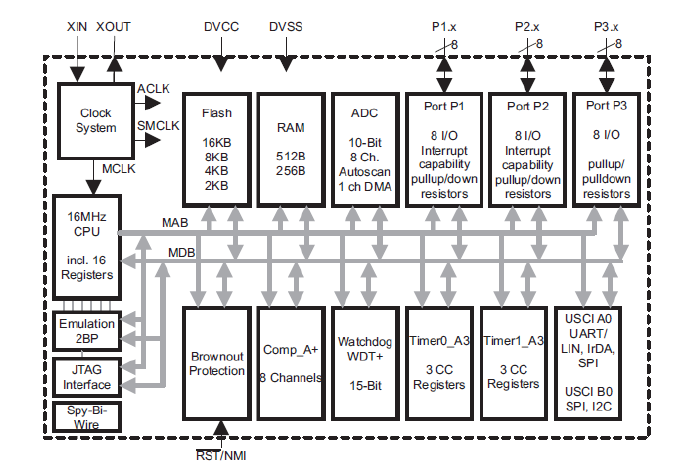
1. **MSP430G2553**

图 5‑1 MSP430G2x53微控制器结构

如图 5‑1所示，此次课程设计主要用到了MSP430G2553内部的定时器功能。

MSP430单片机中Timer\_A定时器就是一种辅助功能强大的定时器，具备捕获和PWM输出等极其有用功能。

MSP430x2xx系列单片机的Timer\_A模块的整体构造如图所示，包括1个16位定时器（Timer Block）和3个捕获比较模块（CCRx）。

1）16位定时器的最大定时值65535，当前计数值被存放在TAR寄存器中。

2）CCRx的捕获模块Caputre由1个输入IO口（CCIx）控制，输入上升沿或下降沿均能触发比较模块动作，捕获发生后的瞬间TAR值被存入TACCRx寄存器。

3）CCRx的比较模块Comparator控制1个输出IO口（TAx）去生成各种脉冲波形。当TAR计数值与预存入TACCRx寄存器的值相等时，比较模块动作，以某种预设规则控制IO电平，生成波形。

4）由于捕获模块Caputre和比较模块Comparator共用了TACCRx寄存

器，捕获Capture的功能是写TACCRx，而比较Comparator的功能是读TACCRx模块，所以捕获和比较不能同时使用。

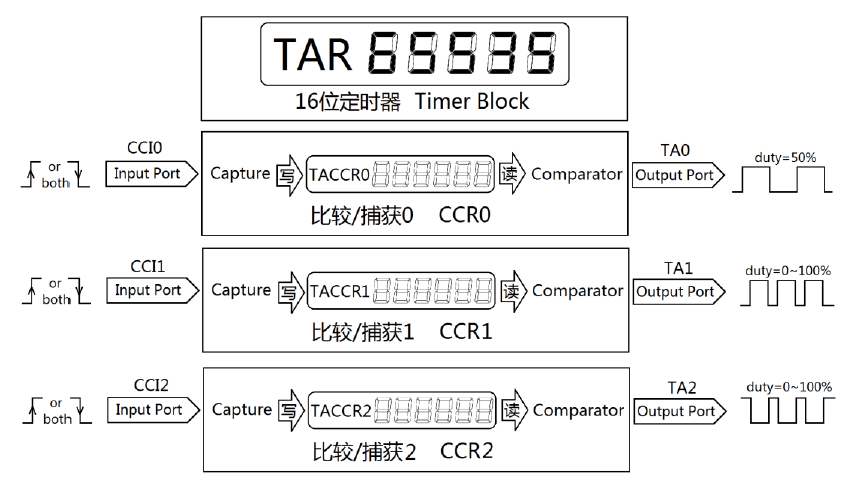


图 5‑2 Timer\_功能框图

**捕获模块**

将CAP设置为1，CCRx工作于捕获模式。主定时器一般设置为连续计数模式，当CCRx检测到CCIx（某带捕获功能的IO口）的电平边沿时，瞬间读取 TAR寄存器的值并写入TACCRx 。CCRx可以选择检测上升沿或下降沿，或者都检测。CCRx用于测定信号脉宽时，只需要分别记录信号上升沿时刻和下降沿时刻，两时刻相减就是脉宽；而测量频率时，连续记录两次上升沿时刻，相减就是周期。

**比较模块**

当CAP=0时，CCRx工作于比较模式。CCR0在比较模式中，将用于设定定时器的周期，所以我们暂时当CCR0“牺牲”了，只讨论CCR1和CCR2的工作情况。当CCR1/2发现TAR的值与TACCR0或它们自己的TACCRx相等时，便会自动改变输出IO口TAx的输出电平，从而生成波形。改变的规则由OUTMODx寄存器决定，共有8种规则。

这8种规则配合主定时器TAR的3种模式（连续计数、增计数、增减计数），可以无需CPU干预生成各种波形。

本次课题，我将P2.1复用为捕获输入引脚，用来捕获光电反射式传感器输出的方波周期。

1. **ST188**

ST188为反射式红外光电传感器。采用高发射功率红外光电二极管和高灵敏度光电晶体管组成。检测距离可调整，采用非接触检测方式。

图 5‑3是光电二极管的外形图，由发射二极管和接收管组成。A、K是红外发射二极管的正负极，C、E是接收管的正负极。因此只要A极接高电平、K极接低电平，红外发射管就能发出红外线。可以在传感器加上外围电路来检测接收管的信号，进而确定是否接收到反射回来的红外线。

图 5‑5图 5‑6为ST188的极限参数和电气特性。

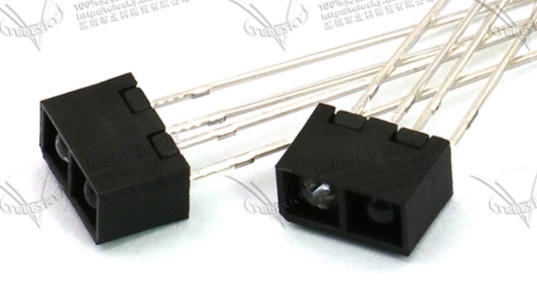


图 5‑3 ST188实物图

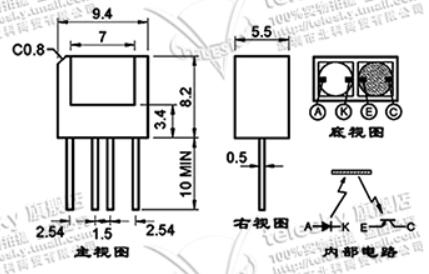
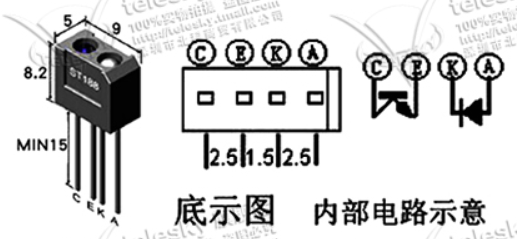


图 5‑4 ST188外形图

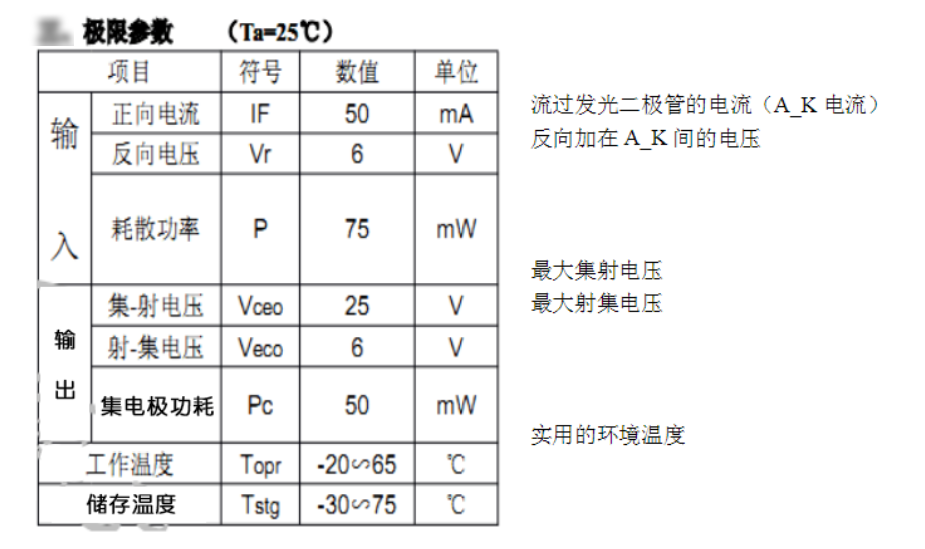


图 5‑5 ST188极限参数

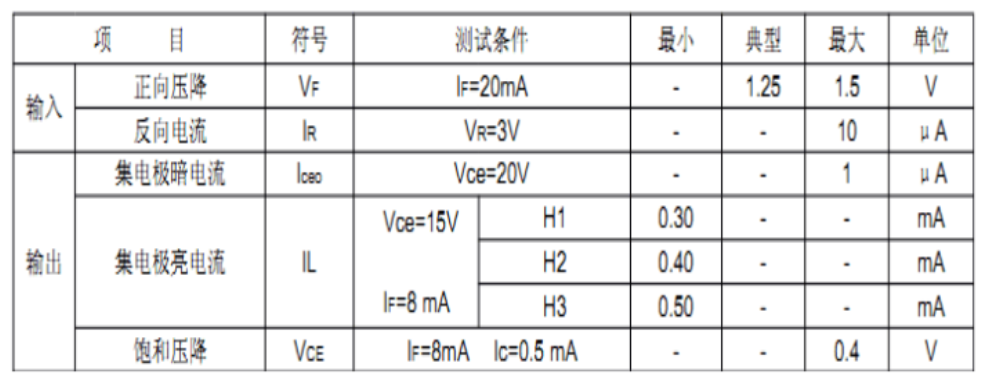


图 5‑6 ST188电气特性

1. **LM393**

LM393 是双电压比较器集成电路。

输出负载电阻能衔接在可允许电源电压范围内的任何电源电压上，不受 Vcc端电压值的限制.此输出能作为一个简单的对地SPS开路(当不用负载电阻没被运用)，输出部分的陷电流被可能得到的驱动和器件的β值所限制.当达到极限电流(16mA)时，输出晶体管将退出而且输出电压将很快上升。

以下是LM393的实物图与内部结构透视图。

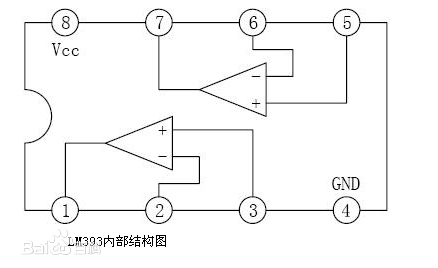
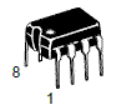


图 5‑7 LM393内部透视图

图 5‑8 LM393实物图

此次课题的光电反射式传感器电路就借助了LM393内部的1个电压比较器实现方波输出。

1. **128段LCD液晶屏**

MSP430G2553通过I2C协议SCL、SDA，对应为P1.6和P1.7去控制扩展版上的TCA6416A芯片输出4个信号CS、WR、RD、DATA，对应的引脚分别为P1.4、P1.6、P1.5、P1.7，控制LCD驱动芯片HT1621，来实现128段LCD的显示。下图 5‑4、图 5‑5为LCD的控制流程和LCD原理图。

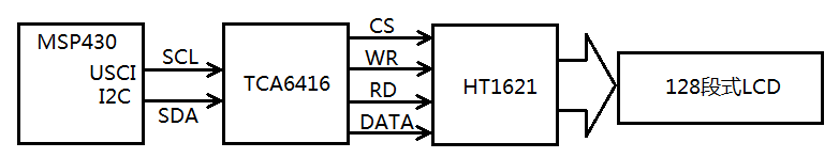


图 5‑9 LCD控制流程

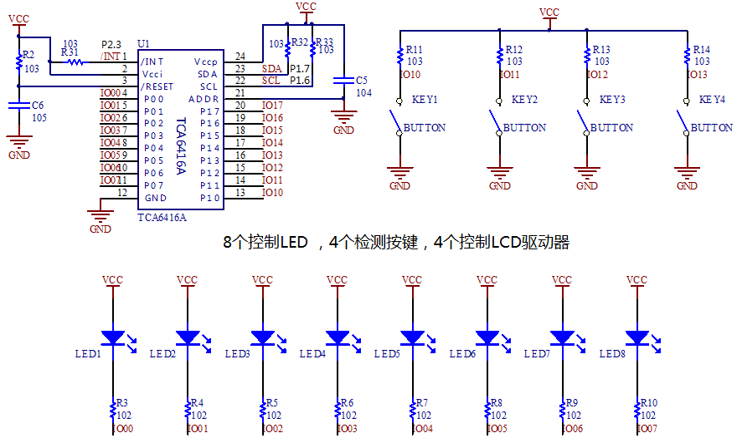
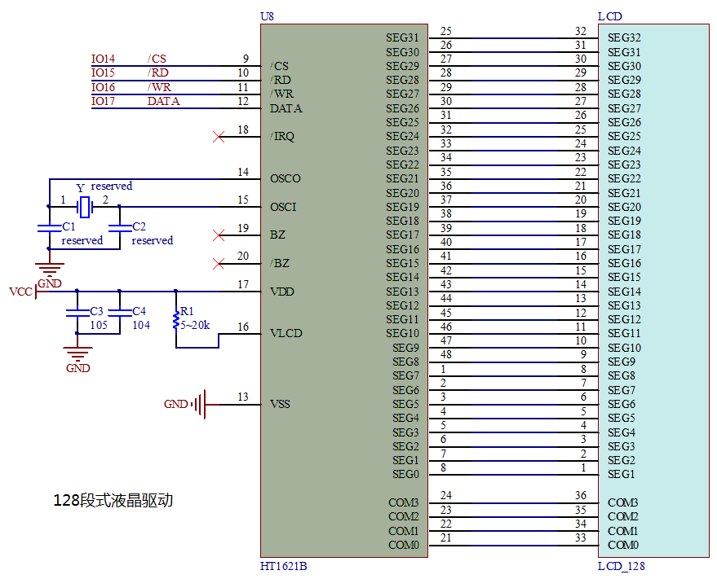


图 5‑10 LCD驱动模块

图 5‑11 基于I2C的IO扩展

## 电路设计

1. **光电传感器模块**

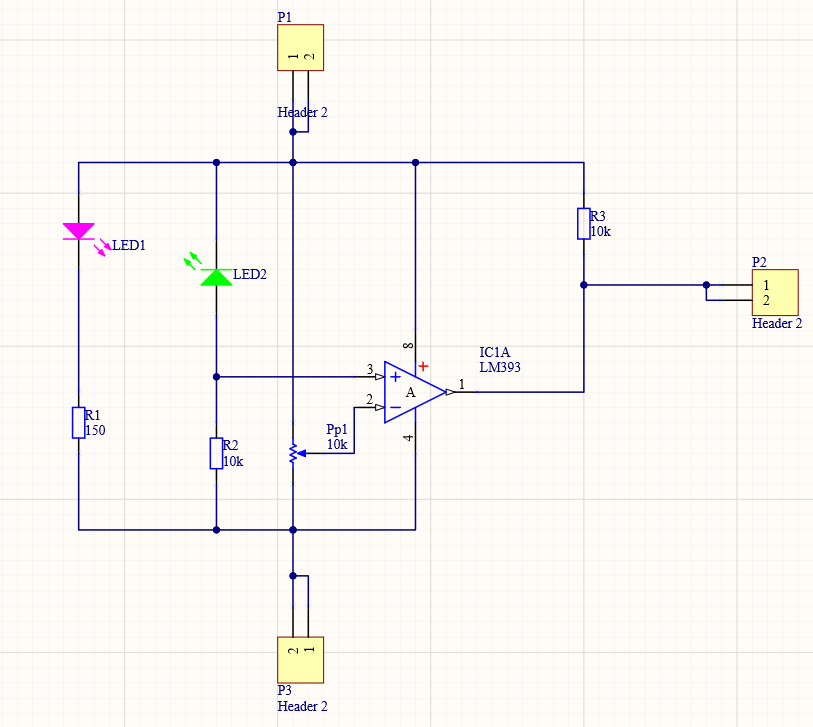


图 5‑12 光电传感器模块

如图 5‑12所示，LED1为红外发射管，LED2为红外接收管。工作原理为当红外接收管LED2接收到红外线时，LED2导通，根据接收到的光强不同，LM393同相输入端电压也会不一样，调节反向输入端的滑动变阻器可以人为规定比较电压。

当前方有障碍物时，LED2接收到反射的红外线，LED2导通，同向输入端的电压大于反向输入端的电压，LM393的1引脚输出高电平。相反当前方没有障碍物时，LED2截至，同向输入端的电压小于反向输入端的电压，LM393的1引脚输出低电平。在测量转速时，前方扇叶或物体的周期遮挡和不遮挡将会使得比较器LM393输出周期固定的方波，将该输出端交给单片机的IO引脚捕获即可。

# 软件设计

## 中断流程图



图 6‑1 TIMER0中断服务函数

如上图 6‑1所示，为定时器TIMER0的中断流程图，当发生定时器TIMER0中断时，进入中断服务函数判断引发的是哪种类型的中断，该定时器用到的是TAIFG溢出中断向量。一旦引发TAIFG溢出中断，则将转速显示标志变量show\_flag=1。由于TIMER0定时器选的时钟源为32768Hz，计数模式为连续模式，所以转速大约每2秒显示一次。



图 6‑2 TIMER1中断服务函数

如图 6‑2所示，为TIMER1的中断服务函数流程图，当发生TIMER1中断时，判断引发的是捕获中断CCIFG还是溢出中断TAIFG。如果是CCIFG中断，并且是第一次上升沿捕获，读取TA1CCR1的值，赋值给变量REdge1，并且将上升沿捕获次数变量count=1，退出LPM0，返回主函数；如果不是第一次上升沿捕获，读取TA1CCR1的值，赋值给变量FEdge2，并且将上升沿捕获次数变量count=0，计算方波的周期，将计数溢出标志变量overflow=0，退出LPM0，返回

主函数。

一旦发生定时器溢出中断TAIFG，将溢出标志overflow累加，退出低功耗模式LPM0，返回主函数。

## 主函数流程图

图 6‑3 主函数流程图

如图 6‑3所示，为此次主程序的流程图，系统上电，系统程序进入初始化，显示个人信息，打开总中断，进入死循环。开始，判断显示标志show\_flag=1，如果show\_flag=1，那么关闭总中断，执行显示函数，将将转速显示标志变量show\_flag=0，关闭总中断，进入低功耗等待中断函数退出低功耗模式LPM0；如果show\_flag=0，那么直接进入低功耗模式LPM0，等待中断函数退出低功耗模式LPM0。

## 显示函数流程图



图 6‑4 显示函数流程图

主函数调用显示函数之前，已经对转速的整数部分和小数部分完成处理，将整数部分保存在变量zhenshu中，将小数部分保存在变量xiaoshu里。调用显示函数时，首先判断整数部分的转速区间，然后再执行不同的取位函数，最后将整数和小数一同显示在128段液晶显示屏上。

# 硬件调试

## 转速测试

图 7‑1 系统上电

如图 7‑1所示，系统上电

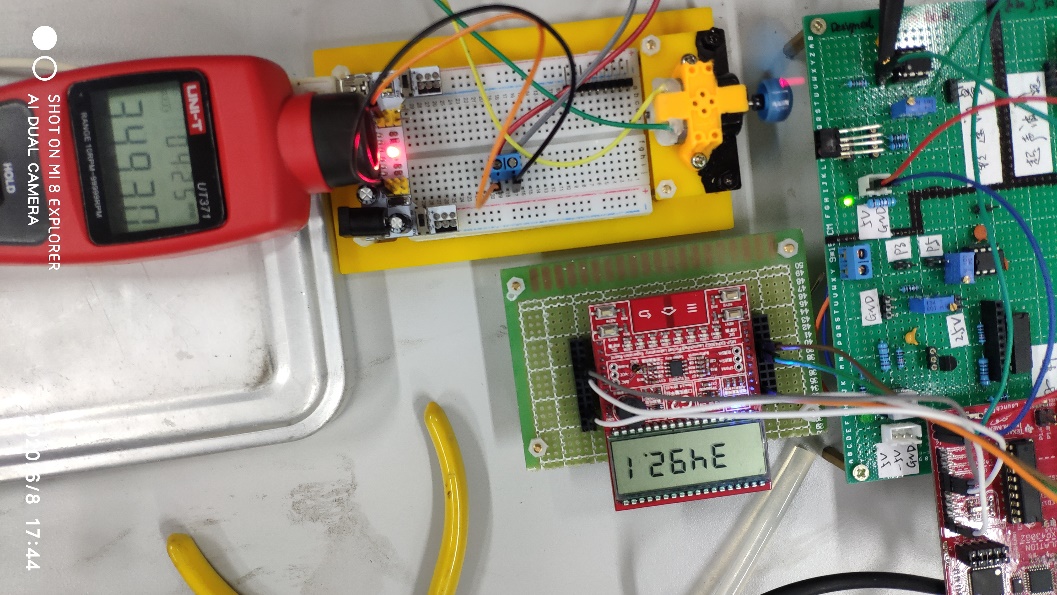


图 7‑2 转速测试1

如图 7‑2所示，为3492rpm测试，LCD显示稳定，比较准确。

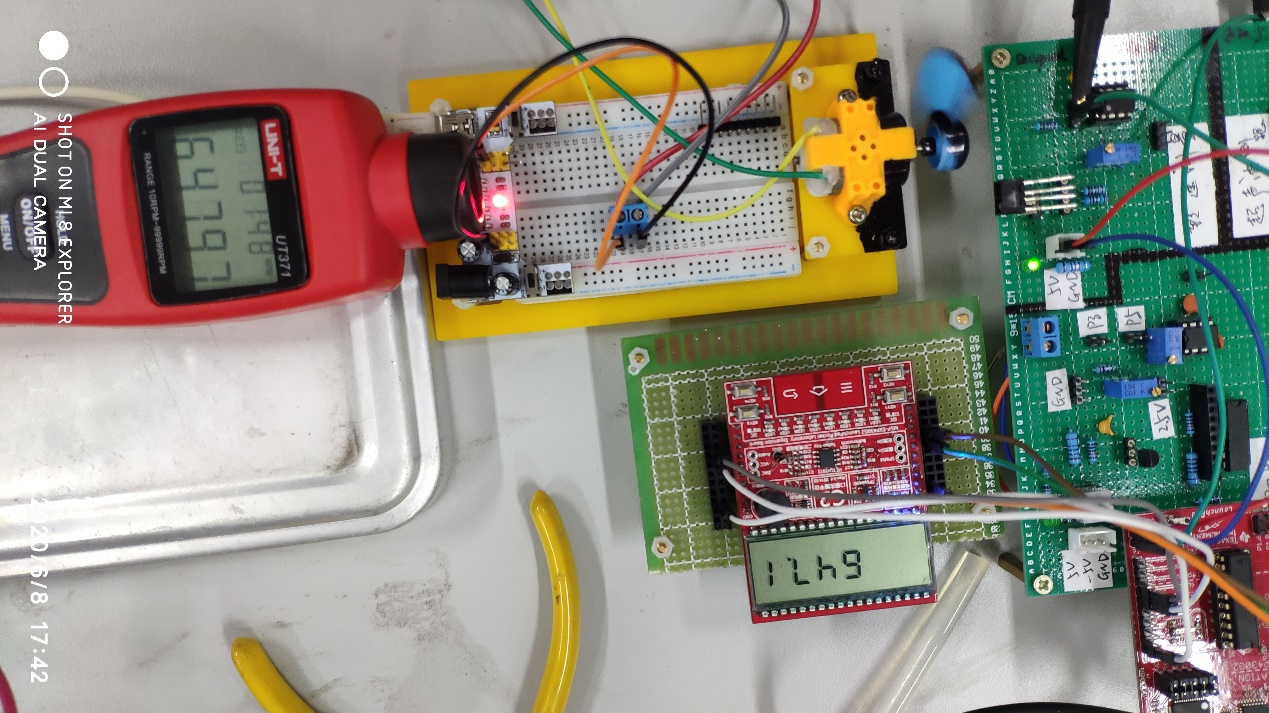


图 7‑3 转速测试2

如图 7‑3所示，为647rpm，LCD显示稳定，比较准确。

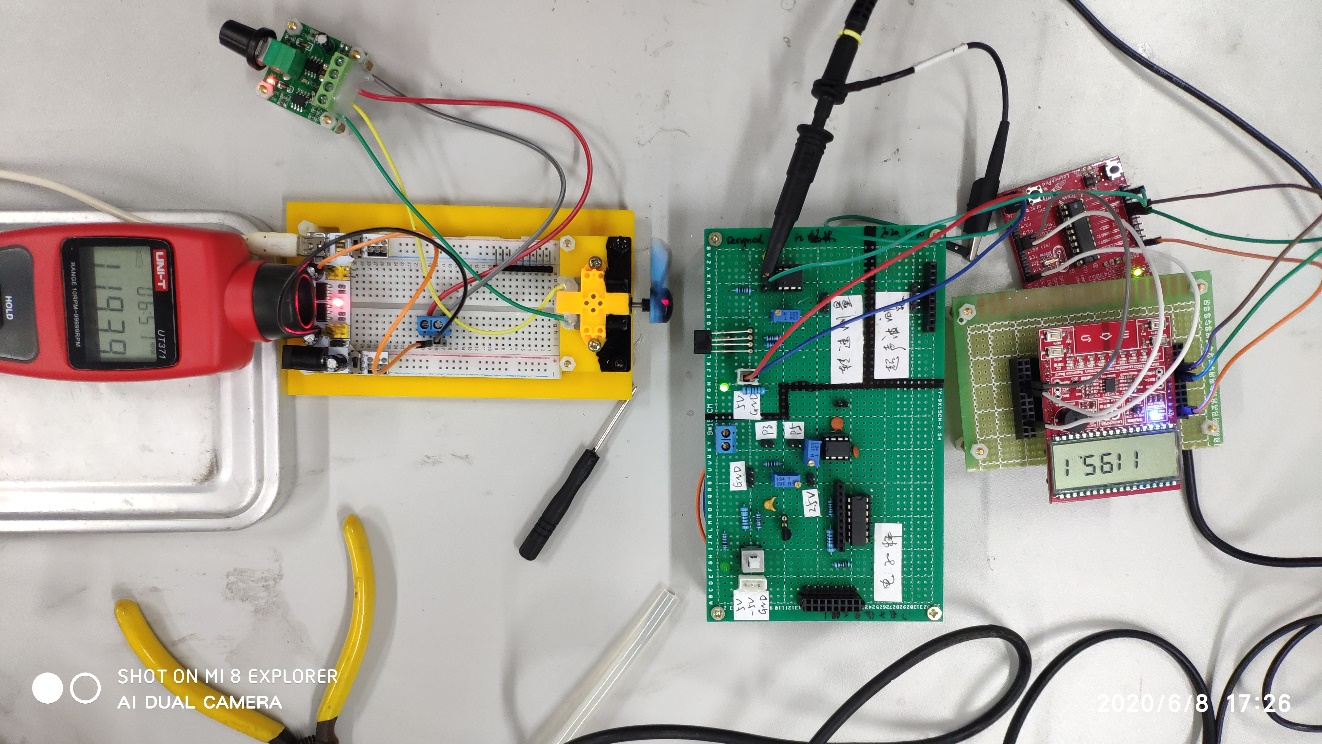


图 7‑5 转速测试3

如图 7‑5所示，为1195rpm，LCD显示稳定，比较准确。

## 数据记录

表格 7‑1 调试数据记录

|  |  |  |
| --- | --- | --- |
| 仪表显示  单位rpm | LCD显示  单位rpm | 误差 |
| 648.0 | 647.1 | -0.13% |
| 1193.9 | 1195.1 | 0.10% |
| 1268.0 | 1269.2 | 0.09% |
| 1318.0 | 1323.4 | 0.41% |
| 1322.1 | 1322.1 | 0.00% |
| 1510.0 | 1511.5 | 0.10% |
| 1827.4 | 1823.8 | -0.20% |
| 1872.0 | 1870.6 | -0.07% |
| 1991.1 | 1989.9 | -0.06% |
| 2303.1 | 2302.2 | -0.04% |
| 2324.6 | 2323.9 | -0.03% |
| 2351.9 | 2351.7 | -0.01% |
| 2407.1 | 2409.4 | 0.10% |
| 2686.0 | 2685.0 | -0.04% |
| 3163.9 | 3160.9 | -0.09% |
| 3510.8 | 3505.6 | -0.15% |
| 3817.9 | 3817.6 | -0.01% |
| 3828.5 | 3825.6 | -0.08% |
| 3837.6 | 3825.0 | -0.33% |
| 4096.0 | 4095.4 | -0.01% |
| 4561.6 | 4560.4 | -0.03% |
| 4727.1 | 4726.1 | -0.02% |
| 4747.3 | 4748.9 | 0.03% |
| 4788.8 | 4795.3 | 0.14% |
| 4991.7 | 4990.0 | -0.03% |
| 5093.4 | 5091.5 | -0.04% |
| 5491.8 | 5491.1 | -0.01% |
| 5649.6 | 5641.6 | -0.14% |
| 5694.5 | 5665.9 | -0.50% |
| 5738.5 | 5732.0 | -0.11% |

实验测量数据如上表所示，可以观察到误差都很小，1%以下。就没有考虑拟合的处理。

# 设计小结

此次的非接触式转速测量表让我对光电传感器有了些许了解。除了翻阅光电的技术手册外，此次的波形捕获，我又一次加深了对MSP430G2553的认识。波形捕获的关键之一在于中断函数的编写，将IO引脚初始化为波形捕获输入后，一旦发生沿的跳变，就会引发CCIFG中断。此次的课题我用到了TIMER1的CCR1作为波形的捕获寄存器，TIMER0作为定时计数器的功能。另外，周期的计算也是波形捕获的重中之重，需要考虑变量的类型、运算溢出、计数溢出等一系列问题。

此次的传感器的课程设计，感谢李老师和季老师的指导与帮助，不管是在硬件电路设计上还是程序设计上他们都给了我特别大的启发。

# 参考文献

[1]毛敏.电机转速测量系统设计[J].山东工业技术,2017(10):189-190.

附件1源程序代码

#include <msp430G2553.h>

#include "LCD\_128.h"

#include "TCA6416A.h"

#include "HT1621.h"

int zhengshu = 0, xiaoshu = 0,show\_flag=0;

char overflow\_times = 0;

float zhuansu = 0;

void LCD\_Init();

void gpio\_init();

void timer0\_init();

void time1\_stop();

void timer1\_init();

void Revolution\_Display();

void display\_my\_info();

void cal\_revolution();

int main(void)

{

WDTCTL = WDTPW + WDTHOLD;

TCA6416A\_Init();

LCD\_Init();

gpio\_init();

timer0\_init();

timer1\_init();

display\_my\_info();

\_EINT();

while (1)

{

if(show\_flag==1)

{

\_DINT();

Revolution\_Display();

show\_flag=0;

\_EINT();

}

LPM0;

}

}

void gpio\_init()

{

P2DIR &= ~BIT1; //P2.1设置为输入

P2SEL |= BIT1; //使用P2.1复用功能

}

void display\_my\_info()

{

LCD\_DisplaySeg(4);

LCD\_DisplaySeg(9);

LCD\_DisplaySeg(10);//显示J

LCD\_DisplayDigit(8, 2);

LCD\_ClearSeg(12);

LCD\_ClearSeg(19);//显示H

LCD\_DisplayDigit(0, 3);

LCD\_ClearSeg(25);

LCD\_ClearSeg(26);

LCD\_ClearSeg(27);//显示L

LCD\_DisplayDigit(1, 4);

LCD\_DisplayDigit(0, 5);

LCD\_DisplayDigit(7, 6);

HT1621\_Reflash(LCD\_Buffer);

\_\_delay\_cycles(1000000);

LCD\_Clear();

LCD\_DisplayDigit(0, 5);

LCD\_DisplayDigit(0, 6);

LCD\_DisplaySeg(\_LCD\_DOT4);

HT1621\_Reflash(LCD\_Buffer);

}

void cal\_revolution()

{

static int REdge1 = 0, REdge2 = 0,Period=0;

static char Count = 0;

const float f=32768;

if (!Count)

{

REdge1 = TA1CCR1;

Count=1;

}

else

{

REdge2 = TA1CCR1;

// time1\_stop();

Count = 0;

Period=REdge2 - REdge1;

zhuansu =(f/(65536.0\*overflow\_times+Period))\*60;

zhengshu = (int)zhuansu; //zhengshu为显示整数，在显示程序中调用

xiaoshu = (zhuansu - zhengshu) \* 10;

overflow\_times = 0;

//TA1R=0;

}

}

void timer0\_init()

{

TA0CTL = TASSEL\_1 + MC\_2 + TACLR + TAIE; // ACLK, Continous up,start,interrupt enable

}

void timer1\_init()

{

TA1CCTL1 = CAP + CM\_1 + CCIE + SCS + CCIS\_0;

TA1CTL |= TASSEL\_1 + MC\_2 + TACLR + TAIE; // SMCLK, Cont Mode; start timer

}

#pragma vector = TIMER0\_A1\_VECTOR

\_\_interrupt void TIMER0\_A0(void)

{

switch(TA0IV)

{

case TA0IV\_NONE: break; // Vector 0: No interrupt

case TA0IV\_TACCR1: break; // Vector 2: TACCR1 CCIFG

case TA0IV\_TACCR2: break; // Vector 4: TACCR2 CCIFG

case TA0IV\_6: break; // Vector 6: Reserved CCIFG

case TA0IV\_8: break; // Vector 8: Reserved CCIFG

case TA0IV\_TAIFG: // Vector 10: TAIFG

show\_flag=1; //-----启用中断服务函数-----

// TA0CTL&=~TAIFG;

break; // Vector 10: TAIFG

default: break;

}

LPM0\_EXIT;

}

#pragma vector = TIMER1\_A1\_VECTOR

\_\_interrupt void TIMER1\_A1\_ISR(void)

{

switch (TA1IV) //AIV的值是在0--10内的偶数时才会执行switch函数内的语句

{

case TA1IV\_NONE:

break; // Vector 0: No interrupt

case TA1IV\_TACCR1: // Vector 2: TACCR1 CCIFG

cal\_revolution();

//\_\_bic\_SR\_register\_on\_exit(LPM0\_bits); // Exit LPM0 on return to main

break;

case TA1IV\_TACCR2:

break; // Vector 4: TACCR2 CCIFG

case TA1IV\_6:

break; // Vector 6: ON\_Perioderved CCIFG

case TA1IV\_8:

break; // Vector 8: ON\_Perioderved CCIFG

case TA1IV\_TAIFG:

overflow\_times += 1;

// TA1CTL &= ~TAIFG;

// \_\_bic\_SR\_register\_on\_exit(LPM0\_bits + GIE);

break;

default:

break;

}

LPM0\_EXIT;

}

void time1\_stop()

{

TA1CTL |= MC\_0 + TACLR; // stop

}

void LCD\_Init()

{

HT1621\_init();

//相关硬件的初始化，其中 I2C 模块的初始化由 TCA6416A 初始化函数在内部完成了， LCD\_128 库函数由 HT1621 初始化函数在内部引用了

//-----显示固定不变的LCD段-----

LCD\_DisplayDigit(5,2);

LCD\_DisplayDigit(8,3);

LCD\_ClearSeg(20);

LCD\_ClearSeg(25);

LCD\_DisplayDigit(6,4);

LCD\_ClearSeg(53);

LCD\_DisplayDigit(6,5);

LCD\_ClearSeg(61);

LCD\_DisplayDigit(0,6);

HT1621\_Reflash(LCD\_Buffer);

\_\_delay\_cycles(1000000);

}

void Revolution\_Display()

{

LCD\_DisplayDigit(LCD\_DIGIT\_CLEAR, 2);

LCD\_DisplayDigit(LCD\_DIGIT\_CLEAR, 3);

LCD\_DisplayDigit(LCD\_DIGIT\_CLEAR, 4);

//-----根据ON\_Period拆分并显示数字-----

if (zhengshu > 999) //1000~9999（4位）

{

LCD\_DisplayDigit(zhengshu / 1000, 2);

LCD\_DisplayDigit((zhengshu / 100) % 10, 3);

LCD\_DisplayDigit((zhengshu / 10) % 10, 4);

LCD\_DisplayDigit(zhengshu % 10, 5);

LCD\_DisplayDigit(xiaoshu, 6);

}

else if (zhengshu > 99) //100~999（3位）

{

LCD\_DisplayDigit(zhengshu / 100, 3);

LCD\_DisplayDigit((zhengshu / 10) % 10, 4);

LCD\_DisplayDigit(zhengshu % 10, 5);

LCD\_DisplayDigit(xiaoshu, 6);

}

else if (zhengshu > 9) //（2位）

{

LCD\_DisplayDigit(zhengshu / 10, 4);

LCD\_DisplayDigit(zhengshu % 10, 5);

LCD\_DisplayDigit(xiaoshu, 6);

}

else

{

LCD\_DisplayDigit(0, 5);

LCD\_DisplayDigit(0, 6);

}

HT1621\_Reflash(LCD\_Buffer); //-----更新缓存，真正显示-----

}