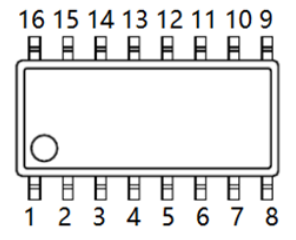
# CS100A超声波测距传感器

## 1.CS100A介绍

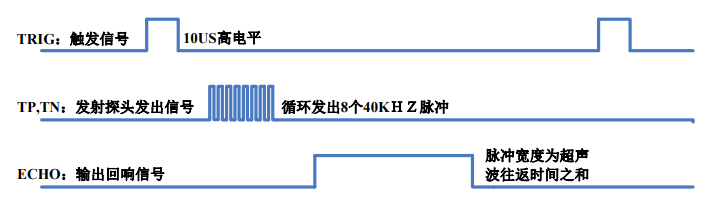
CS100A是一款工业级超声波测距芯片，内部集成了超声波发射，接收电路，数字处理电路。CS100A需要配合400KHz的开放式超声波探头，测距范围在8m左右，探测精度0.1cm+1%,测量盲区小于2cm。引脚功能如下图所示：



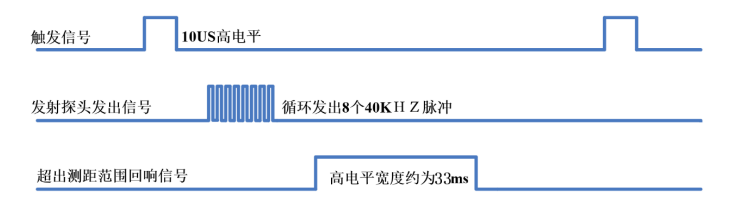


## 2.工作原理

在芯片TRIG引脚输入一个10us以上的高电平（一般建议50us左右），TP,TN管脚便可发出8个40KHz的超声波脉冲，然后RP，RN检测到回波信号（正常情况），ECHO引脚输出高电平，然后根据ECHO引脚输出高电平的持续时间可以计算距离值，计算公式为：距离=（高电平持续时间\*340m/s）/2。测距时序如下图所示：



当测量距离超过测量范围时，CS100A仍会通过ECHO引脚输出高电平，高电平持续时间约为33ms,如下图所示



最坏情况下，被测物体超出测量范围，此时ECHO高电平持续时间最长，为33ms,所以一般测量周期应大于33ms(可取50ms)。

## 3.程序实现

本例程使用的MCU为沁恒的CH32V307VCT6，频率72Mhz。

首先我们需要配置两个GPIO引脚，一个连接TRIG引脚，用于发出50us的触发信号，可配置为推挽输出；一个连接ECHO引脚，用于接收高电平信号，可配置为浮空输入，并且该引脚还需要配置外部中断，以便检测到高电平后立刻开始计算时间。引脚初始化程序如下：

//GPIO初始化

    GPIO\_InitTypeDef GPIO\_InitStructure = {0};

    EXTI\_InitTypeDef EXTI\_InitStructure = {0};

    NVIC\_InitTypeDef NVIC\_InitStructure = {0};

    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO|CS100A\_GPIO\_CLK, ENABLE);

    GPIO\_InitStructure.GPIO\_Pin = CS100A\_TRIG\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

    GPIO\_Init(CS100A\_GPIO\_PORT, &GPIO\_InitStructure);

    GPIO\_InitStructure.GPIO\_Pin = CS100A\_ECHO\_PIN;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

    GPIO\_Init(CS100A\_GPIO\_PORT, &GPIO\_InitStructure);

    GPIO\_ResetBits(CS100A\_GPIO\_PORT, CS100A\_TRIG\_PIN);//TRIG引脚默认低电平，输入一个10us以上的高电平后，便可发出超声波脉冲信号

    GPIO\_ResetBits(CS100A\_GPIO\_PORT, CS100A\_ECHO\_PIN);

    //ECHO引脚中断初始化

    GPIO\_EXTILineConfig(CS100A\_PORTSOURCE , CS100A\_PINSOURCE);

    EXTI\_InitStructure.EXTI\_Line = CS100A\_EXTILINE; // 外部中断线2

    EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt; // 配置为外部中断

    EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Rising\_Falling;

    EXTI\_InitStructure.EXTI\_LineCmd = ENABLE; // 使能外部中断

    EXTI\_Init(&EXTI\_InitStructure);

    NVIC\_InitStructure.NVIC\_IRQChannel = CS100A\_EXTI\_IRQ; //

    NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

    NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

    NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; // 使能中断

    NVIC\_Init(&NVIC\_InitStructure);

需要注意的是这里外部中断配置为上升沿和下降沿都会触发中断，这与接下来的高电平持续时间计算方式有关。

GPIO初始化后，还需要初始话一个定时器用来计算ECHO高电平持续时间，定时器初始化如下所示：

   //计时定时器初始化

    TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

    RCC\_APB1PeriphClockCmd(CS100A\_TIM\_CLK, ENABLE);

    TIM\_TimeBaseStructure.TIM\_Period = (10-1);                 // 自动重装载寄存器的值，累计TIM\_Period+1个频率后产生一个更新或者中断

    TIM\_TimeBaseStructure.TIM\_Prescaler = 72 - 1;               // 定时器频率为1MHz

    TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;                // 分频因子

    TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up; // 向上计时

    TIM\_TimeBaseInit(CS100A\_TIM, &TIM\_TimeBaseStructure);

    NVIC\_InitStructure.NVIC\_IRQChannel = CS100A\_TIM\_IRQ ;

    NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

    NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1;

    NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

    NVIC\_Init(&NVIC\_InitStructure);

    /\* 定时器中断开启 \*/

    TIM\_ClearFlag(CS100A\_TIM,TIM\_IT\_Update);         // 清除计时中断标志位

    TIM\_ITConfig(CS100A\_TIM, TIM\_IT\_Update, ENABLE);   // 开启计时中断

    TIM\_SetCounter(CS100A\_TIM, 0);                     // 设置初始计数为0

    TIM\_Cmd(CS100A\_TIM,DISABLE);                       // 使能

这里定时器分频值为72，也就是定时器1us计一个数，自动重装载值配置为10，同时该定时器也配置了定时器中断，也就是说进入1次定时器中断表示计时10us,再加上定时器1us/次的计数值，就可以得出精度较高的测量时间。

初始化完成后，需要给TRIG引脚提供一个持续10us以上的高电平，开始测量，这里我们延时50us。程序如下：

    GPIO\_SetBits(CS100A\_GPIO\_PORT, CS100A\_TRIG\_PIN);

    Delay\_Us(50);

    GPIO\_ResetBits(CS100A\_GPIO\_PORT, CS100A\_TRIG\_PIN);

开始测量后，我们需要检测ECHO引脚的高电平持续时间，然后计算距离。这我们使用一个ECHO引脚外部中断和定时器中断结合来处理，当ECHO引脚高电平产生时，触发上升沿中断，在中断函数中设定定时器计数值为0，使能定时器。定时器每10us进入一次定时器中断，在定时器中断里自增一次中断计数，记录中断触发次数。等ECHO引脚低电平产生时，触发下降沿中断，在中断函数中开始计算距离。此时ECHO高电平持续时间就等于定时器中断触发次数乘以10us加上定时器计数（每1us计1次），带入距离计算公式，就可以得出测量距离。中断函数程序如下所示：

//超声波传感器

void CS100A\_EXTI\_IRQHandler(void)

{

    if(EXTI\_GetITStatus(CS100A\_EXTILINE)!=RESET)

    {

        EXTI\_ClearITPendingBit(CS100A\_EXTILINE);

        if(GPIO\_ReadInputDataBit(CS100A\_GPIO\_PORT, CS100A\_ECHO\_PIN)==1)

        {

            TIM\_SetCounter(CS100A\_TIM, 0);

            TIM\_Cmd(CS100A\_TIM, ENABLE);

        }

        else

        {

            gCS100A\_Distance = (float)(gCS100ACount\*10+TIM\_GetCounter(CS100A\_TIM))/1000\*34.0/2.0;//34cm/ms,gCS100A\_Distance全局变量

            gCS100ACount = 0;

            TIM\_Cmd(CS100A\_TIM, DISABLE);

        }

    }

}

//超声波传感器计时

void CS100A\_TIM\_IRQHandler(void)

{

    if ( TIM\_GetITStatus( CS100A\_TIM, TIM\_IT\_Update) != RESET )

    {

        TIM\_ClearFlag(CS100A\_TIM , TIM\_FLAG\_Update);

        gCS100ACount++;

    }

}

需要注意的是，CH32V307使用的是RSICV内核，沁恒的RSICV内核处理器在编译时，需要通过特殊的关键字标识函数为中断函数，否则会被编译器视作普通函数处理，中断执行完成之后，中断执行之前保存的信息将不会被返回，中断会被认为一个正常程序继续执行下去，但中断之后并没有程序，从而导致程序出现类似跑飞状况。官方对该题的解释：<https://www.wch.cn/bbs/thread-84071-1.html>

解决方式就是在中断函数的声明和定义中加入以下关键字

void XXXX\_IRQHandler(void) **attribute**((interrupt(“WCH-Interrupt-fast”)))；

**完整驱动代码如下：**

**CS100A.h**

#ifndef \_\_CS100A\_H

#define \_\_CS100A\_H

#include "debug.h"

#include "user.h"

#define CS100A\_GPIO\_PORT        GPIOD

#define CS100A\_GPIO\_CLK         RCC\_APB2Periph\_GPIOD

#define CS100A\_TRIG\_PIN         GPIO\_Pin\_5

#define CS100A\_ECHO\_PIN         GPIO\_Pin\_6

#define CS100A\_PORTSOURCE       GPIO\_PortSourceGPIOD

#define CS100A\_PINSOURCE        GPIO\_PinSource6

#define CS100A\_EXTILINE         EXTI\_Line6

#define CS100A\_EXTI\_IRQ         EXTI9\_5\_IRQn

#define CS100A\_EXTI\_IRQHandler  EXTI9\_5\_IRQHandler

#define CS100A\_TIM\_CLK          RCC\_APB1Periph\_TIM3

#define CS100A\_TIM              TIM3

#define CS100A\_TIM\_IRQ          TIM3\_IRQn

#define CS100A\_TIM\_IRQHandler   TIM3\_IRQHandler

void CS100A\_Init(void);

void CS100A\_GET(void);//获取距离存到全局变量中

#endif

**CS100A.c**

#include "CS100A.h"

#include "ch32v30x\_it.h"

//超声波传感器

void CS100A\_EXTI\_IRQHandler(void) \_\_attribute\_\_((interrupt("WCH-Interrupt-fast")));

void TIM3\_IRQHandler(void) \_\_attribute\_\_((interrupt("WCH-Interrupt-fast")));

void CS100A\_Init(void)

{

    //GPIO初始化

    GPIO\_InitTypeDef GPIO\_InitStructure = {0};

    EXTI\_InitTypeDef EXTI\_InitStructure = {0};

    NVIC\_InitTypeDef NVIC\_InitStructure = {0};

    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO|CS100A\_GPIO\_CLK, ENABLE);

    GPIO\_InitStructure.GPIO\_Pin = CS100A\_TRIG\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

    GPIO\_Init(CS100A\_GPIO\_PORT, &GPIO\_InitStructure);

    GPIO\_InitStructure.GPIO\_Pin = CS100A\_ECHO\_PIN;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

    GPIO\_Init(CS100A\_GPIO\_PORT, &GPIO\_InitStructure);

    GPIO\_ResetBits(CS100A\_GPIO\_PORT, CS100A\_TRIG\_PIN);//TRIG引脚默认低电平，输入一个10us以上的高电平后，便可发出超声波脉冲信号

    GPIO\_ResetBits(CS100A\_GPIO\_PORT, CS100A\_ECHO\_PIN);

    //ECHO引脚中断初始化

    GPIO\_EXTILineConfig(CS100A\_PORTSOURCE , CS100A\_PINSOURCE);

    EXTI\_InitStructure.EXTI\_Line = CS100A\_EXTILINE; // 外部中断线2

    EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt; // 配置为外部中断

    EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Rising\_Falling;

    EXTI\_InitStructure.EXTI\_LineCmd = ENABLE; // 使能外部中断

    EXTI\_Init(&EXTI\_InitStructure);

    NVIC\_InitStructure.NVIC\_IRQChannel = CS100A\_EXTI\_IRQ; //

    NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

    NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

    NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; // 使能中断

    NVIC\_Init(&NVIC\_InitStructure);

   //计时定时器初始化

    TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

    RCC\_APB1PeriphClockCmd(CS100A\_TIM\_CLK, ENABLE);

    TIM\_TimeBaseStructure.TIM\_Period = (10-1);                 // 自动重装载寄存器的值，累计TIM\_Period+1个频率后产生一个更新或者中断

    TIM\_TimeBaseStructure.TIM\_Prescaler = 72 - 1;               // 定时器频率为1MHz

    TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;                // 分频因子

    TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up; // 向上计时

    TIM\_TimeBaseInit(CS100A\_TIM, &TIM\_TimeBaseStructure);

    NVIC\_InitStructure.NVIC\_IRQChannel = CS100A\_TIM\_IRQ ;

    NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

    NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1;

    NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

    NVIC\_Init(&NVIC\_InitStructure);

    /\* 定时器中断开启 \*/

    TIM\_ClearFlag(CS100A\_TIM,TIM\_IT\_Update);         // 清除计时中断标志位

    TIM\_ITConfig(CS100A\_TIM, TIM\_IT\_Update, ENABLE);   // 开启计时中断

    TIM\_SetCounter(CS100A\_TIM, 0);                     // 设置初始计数为0

    TIM\_Cmd(CS100A\_TIM,DISABLE);                       // 使能

}

//测量物体距离太近的话，不能保证接收模块能够接收到发射模块的信息，从而导致测距失败。

void CS100A\_GET(void)

{

    GPIO\_SetBits(CS100A\_GPIO\_PORT, CS100A\_TRIG\_PIN);

    Delay\_Us(50);

    GPIO\_ResetBits(CS100A\_GPIO\_PORT, CS100A\_TRIG\_PIN);

}

//超声波传感器

void CS100A\_EXTI\_IRQHandler(void)

{

    if(EXTI\_GetITStatus(CS100A\_EXTILINE)!=RESET)

    {

        EXTI\_ClearITPendingBit(CS100A\_EXTILINE);

        if(GPIO\_ReadInputDataBit(CS100A\_GPIO\_PORT, CS100A\_ECHO\_PIN)==1)

        {

            TIM\_SetCounter(CS100A\_TIM, 0);

            TIM\_Cmd(CS100A\_TIM, ENABLE);

        }

        else

        {

            gCS100A\_Distance = (float)(gCS100ACount\*10+TIM\_GetCounter(CS100A\_TIM))/1000\*34.0/2.0;//34cm/ms,gCS100A\_Distance全局变量

            gCS100ACount = 0;

            TIM\_Cmd(CS100A\_TIM, DISABLE);

        }

    }

}

//超声波传感器计时

void CS100A\_TIM\_IRQHandler(void)

{

    if ( TIM\_GetITStatus( CS100A\_TIM, TIM\_IT\_Update) != RESET )

    {

        TIM\_ClearFlag(CS100A\_TIM , TIM\_FLAG\_Update);

        gCS100ACount++;

    }

}