# 23.HC_SR04 超声波模块使用实验例程

## 23.1. 实验目的

使用盘古 PGX-Nano 开发板驱动 HC_SR04 超声波模块测距并通过串口打印输出

## 23.2. 实验简介

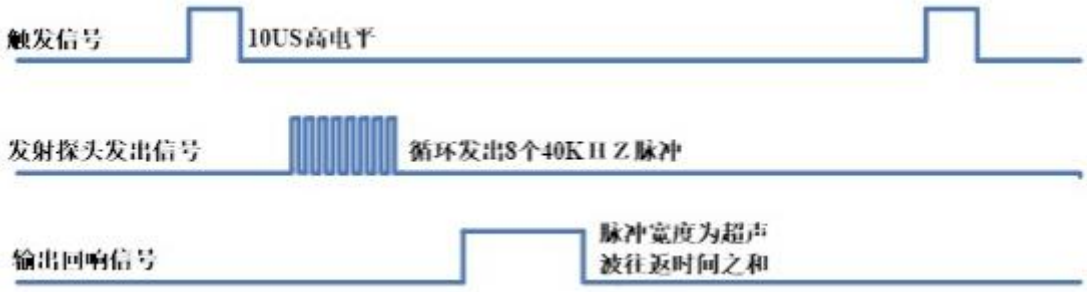
### 23.2.1. 超声波测距原理

超声波测距是一种利用超声波在空气中的传播时间来测量距离的方法。只要知道超声波的传播速度以及发射超声波与接收到反射声波的时间就能计算出物体的距离。其具体操作过程如下（不考虑温度、压强对声波传输的影响）：1.发射超声波 超声波测距模块首先通过一个超声波换能器（通常是压电陶瓷或压电薄膜）发射出一束超声波脉冲。这些脉冲在空气中以一定的速度传播。2.接收反射波 当超声波脉冲被目标物体反射后，返回的反射波被同一个或另一个超声波换能器接收。测距模块记录从发射到接收反射波的时间。3.计算距离 根据超声波的传播时间和传播速度，只需要将（传播时间×声速）/2 就可以得到目标物体与测距模块之间的距离。这里的传播时间是从发射到接收到反射波的总时间，除以 2 是因为超声波往返的总时间包括去程和返程。

### 23.2.2. HC_SR04 模块介绍

HC-SR04 超声波测距模块是一种经济实用的非接触式距离测量设备，广泛用于各种需要精确测距的应用，如机器人避障、汽车倒车雷达等。该模块包含一个超声波发射器和接收器，通过四个引脚（VCC、Trig、Echo、GND）与微控制器连接。工作时，需要微控制器向 Trig 引脚发送一个 10 微秒的高电平脉冲，触发模块发射超声波脉冲。模块接收到反射波后，Echo 引脚输出一个与超声波往返时间成正比的高电平脉冲，微控制器需要通过测量 Echo 引脚高电平脉冲的持续时间，利用距离=（传播时间×声速）/2 计算出目标距离。

其具体控制时序如下：




我们使用 PGX-Nano 开发板作为微控制器，首先我们向 HC-SR04 模块的 trig 引脚发送一个持续 10us 的触发信号，HC-SR04 收到我们给出的触发信号之后，会使用超声波探头发射 8 个 40khz 的脉冲。当测距结束后，HC-SR04 会通过 echo 引脚发送一个高电平信号，高电平持续时间就是声波往返物体经历的时间。我们利用这段时间就可以计算出物体到超声波模块的距离。

## 23.3. 程序设计

超声波驱动模块端口如下：

| 端口 | I/O | 位宽 | 描述 |
|---|---|---|---|
| clk | input | 1 | 50Mhz 时钟 |
| rst_n | input | 1 | 复位信号 |
| cap_flag | input | 1 | 触发采集信号（开始一次测距） |
| distance | output | 8 | 计算得到的距离（单位厘米） |
| data_vld | output | 1 | distance 的有效标志 |
| data_vld_d | output | 1 | 个、十、百、千位的数据有效标志 |
| s_g | output | 4 | distance 的个位 |
| s_s | output | 4 | distance 的十位 |
| s_b | output | 4 | distance 的百位 |
| s_q | output | 4 | distance 的千位 |
| echo | input | 1 | 超声波模块输入 |
| trig | output | 1 | 输出给超声波模块的触发信号 |

驱动部分代码如下，完整源码请查看 demo 源文件

```
always @(posedge clk or negedge rst_n)
   if(!rst_n)
      cnt_en <= 1'b0;
   else if(cap_flag)
```

```
        cnt_en <= 1'b1;
    else if(cnt_trig == 10'd500 -1)
        cnt_en <= 1'b0;

always @(posedge clk or negedge rst_n)
    if(!rst_n)
        cnt_trig <= 10'b0;
    else if(cnt_trig == 10'd500 -1)
        cnt_trig <= 10'b0;
    else if(cnt_en)
        cnt_trig <= cnt_trig + 1'b1;
always @(posedge clk or negedge rst_n)
    if(!rst_n)
        cnt_1us <= 7'b0;
    else if((cnt_1us==MAX_CNT-1)||(cap_flag))
          cnt_1us <= 7'b0;
    else if(echo_syn3)
          cnt_1us <= cnt_1us + 1'b1;
always @(posedge clk or negedge rst_n)
    if(!rst_n)
        cnt_us_num <= 13'b0;
    else if((cap_flag))
          cnt_us_num <= 13'b0;
    else if((echo_syn3)&&(cnt_1us==MAX_CNT-1))
          cnt_us_num <= cnt_us_num + 1'b1;
always @(posedge clk or negedge rst_n)
    if(!rst_n)begin
        distance<= 8'b0;
        data_vld<= 1'b0;
    end
    else if(echo_fall)begin
        distance <= (cnt_us_num * 17)/1000;
        data_vld <= 1'b1;
    end
    else
        data_vld<= 1'b0;
always @(posedge clk or negedge rst_n) begin
    if (rst_n == 1'b0) begin
        s_g <= 0;
        s_s <= 0;
        s_b <= 0;
        s_q <= 0;
        data_vld_d<= 1'b0;
    end else begin
```

```
      s_g <= distance % 10;
      s_s <= (distance / 10) % 10;
      s_b <= (distance / 100) % 10;
      s_q <= (distance / 1000) % 10;
      data_vld_d<= data_vld;
   end
end
```

这个模块的功能是通过控制 HC-SR04 超声波测距模块的触发信号和接收反射信号，测量目标物体的距离并将结果输出。模块根据触发标志信号 cap_flag 启动测距过程。在代码 62 行，当 cap_flag 有效时，将 cnt_en 拉高，开始 10us 的触发计数器 cnt_trig 的计数当计数到 499 时（模块时钟是 50Mhz，一个时钟周期是 20ns，10us 的时间就是 500 个时钟周期，由于计数器是从 0 开始计数，所以计数最大值是 499），清零并拉低 cnt_en。这个 cnt_en 信号持续了 10us，将其作为超声波测距模块的触发信号输出。

由于超声波模块输出的信号时钟与我们开发板的时钟不同步，我们声明 echo_syn1、echo_syn2、echo_syn3 使用多级寄存器打拍的方式对 echo 信号进行时钟同步。当 echo_syn3 有效时，启动微秒计数器 cnt_1us 对 echo 高电平持续时间进行计数。同时使用 cnt_us_num 对高电平持续时间有几微秒进行记录。当捕获到 echo 的下降沿之后，结合 cnt_us_num 的值对距离进行计算。通过以下这行可以实现：　　distance <= (cnt_us_num * 17)/1000　　其实是 distance <= (cnt_us_num * 340)/2/1000;模块中对其进行了适当的简化，除以 1000 是将单位转换为厘米。最终，模块将距离值分解为个位、十位、百位和千位，通过输出寄存器输出。

串口发送代码如下：

```
// 串口发送数据控制
always @(posedge clk or negedge rst_n) begin
   if (!rst_n) begin
      uart_tx_req <= 1'b0;
      uart_tx_data <= 8'b0;
      ascii_table[0] = 8'd48;                                    //'0'
      ascii_table[1] = 8'd49;                                    //'1'
      ascii_table[2] = 8'd50;                                    //'2'
      ascii_table[3] = 8'd51;                                    //'3'
      ascii_table[4] = 8'd52;                                    //'4'
      ascii_table[5] = 8'd53;                                    //'5'
      ascii_table[6] = 8'd54;                                    //'6'
      ascii_table[7] = 8'd55;                                    //'7'
      ascii_table[8] = 8'd56;                                    //'8'
      ascii_table[9] = 8'd57;                                    //'9'
   end else if (work_en && !uart_tx_busy) begin
```

```
      case (tx_byte_cnt)                                          // 根据当前字节计数
器发送数据
            7'd0:  uart_tx_data <= STR[191:184];                  // "超"
            7'd1:  uart_tx_data <= STR[183:176];                  // "超"
            7'd2:  uart_tx_data <= STR[175:168];                  // "声"
            7'd3:  uart_tx_data <= STR[167:160];                  // "声"
            7'd4:  uart_tx_data <= STR[159:152];                  // "波"
            7'd5:  uart_tx_data <= STR[151:144];                  // "波"
            7'd6:  uart_tx_data <= STR[143:136];                  // "模"
            7'd7:  uart_tx_data <= STR[135:128];                  // "模"
            7'd8:  uart_tx_data <= STR[127:120];                  // "块"
            7'd9:  uart_tx_data <= STR[119:112];                  // "块"
            7'd10: uart_tx_data <= STR[111:104];                  // "测"
            7'd11: uart_tx_data <= STR[103:96];                   // "测"
            7'd12: uart_tx_data <= STR[95:88];                    // "距"
            7'd13: uart_tx_data <= STR[87:80];                    // "距"
            7'd14: uart_tx_data <= STR[79:72];                    // "值"
            7'd15: uart_tx_data <= STR[71:64];                    // "值"
            7'd16: uart_tx_data <= STR[63:56];                    // "为"
            7'd17: uart_tx_data <= STR[55:48];                    // "为"
            7'd18: uart_tx_data <= STR[47:40];                    // "："
            7'd19: uart_tx_data <= STR[39:32];                    // "："
            7'd20: uart_tx_data <= ascii_table[s_q];              // 测量距离的个位
            7'd21: uart_tx_data <= ascii_table[s_b];              // 测量距离的十位
            7'd22: uart_tx_data <= ascii_table[s_s];              // 测量距离的百位
            7'd23: uart_tx_data <= ascii_table[s_g];              // 测量距离的千位
            7'd24: uart_tx_data <= STR[31:24];                    // "厘"
            7'd25: uart_tx_data <= STR[23:16];                    // "厘"
            7'd26: uart_tx_data <= STR[15:8];                     // "米"
            7'd27: uart_tx_data <= STR[7:0];                      // "米"
            7'd28: uart_tx_data <= 8'h0A;                         // 换行符
            default: uart_tx_data <= 8'b0;
      endcase
      uart_tx_req <= 1'b1;                                        // 拉高请求信号
   end else begin
      uart_tx_req <= 1'b0;                                        // 请求信号拉高仅一
个时钟周期
   end
end
```

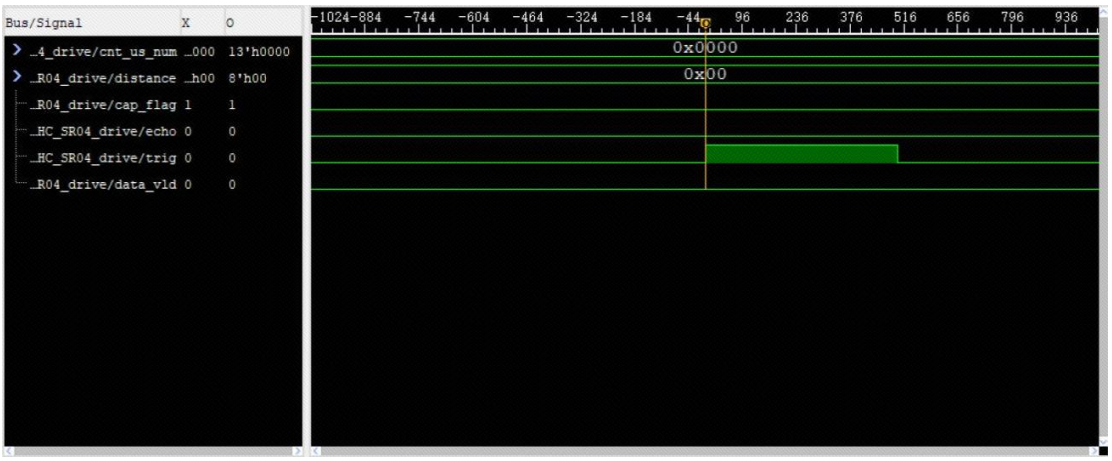
模块中初始化了一个 ascii_table 用来储存阿拉伯数字 0~9 的 ASCII 值，将超声波测距得到的数据分为个位、十位、百位、千位在 ascii_table 索引到对应的 ASCII 值进行发送。

## 23.4. 代码仿真

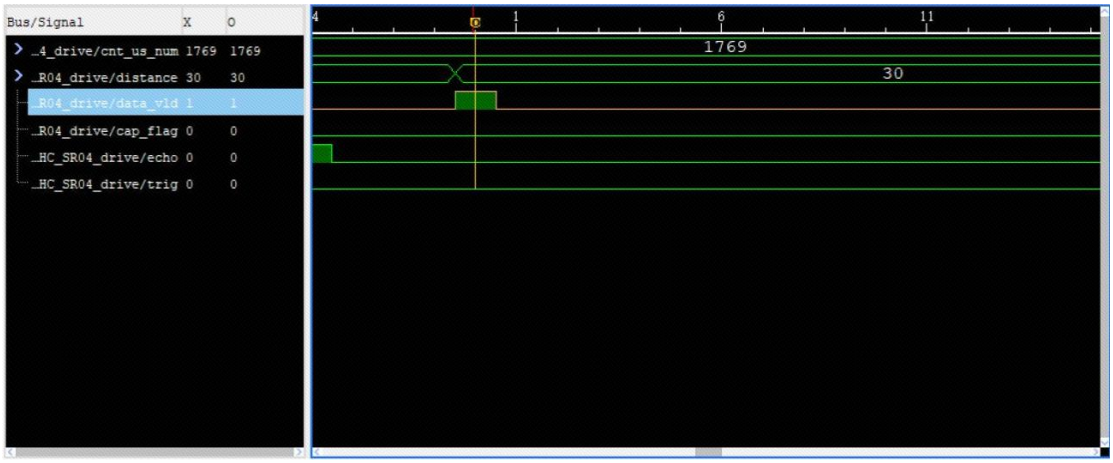
由于本次实验涉及到与 HC_SR04 超声波测距模块的交互，编写测试代码模拟 HC_SR04 超声波测距模块的答复比较麻烦，而且不能贴近真实的模块真实的测距情景。所以我们可以使用 PDS 在线 debug 的功能对 HC_SR04 驱动模块进行验证。

我们在 HCSR04 驱动模块代码中对关键信号打上标记（/ synthesis PAP*MARK_DEBUG="true" */），使用在线 debug 工具对模块进行分析。 我们将采样触发信号 cap_flag、计算得到的距离 distance、输出的数据有效信号 data_vld、超声波测距模块输入的回响信号 echo、输入给超声波测距模块的触发信号 trig、回响信号的微秒计数器 cnt_us_num 打上标记，进行分析。

我们将触发模式设置为 cap_flag 信号高电平触发模式，按下按键，cap_flag 被拉高（下图中 cap_flag 的值为 1，只是被标志线遮挡住了），同时触发信号 trig 正确的拉高了。说明已经向超声波测距模块发送了测距触发信号。



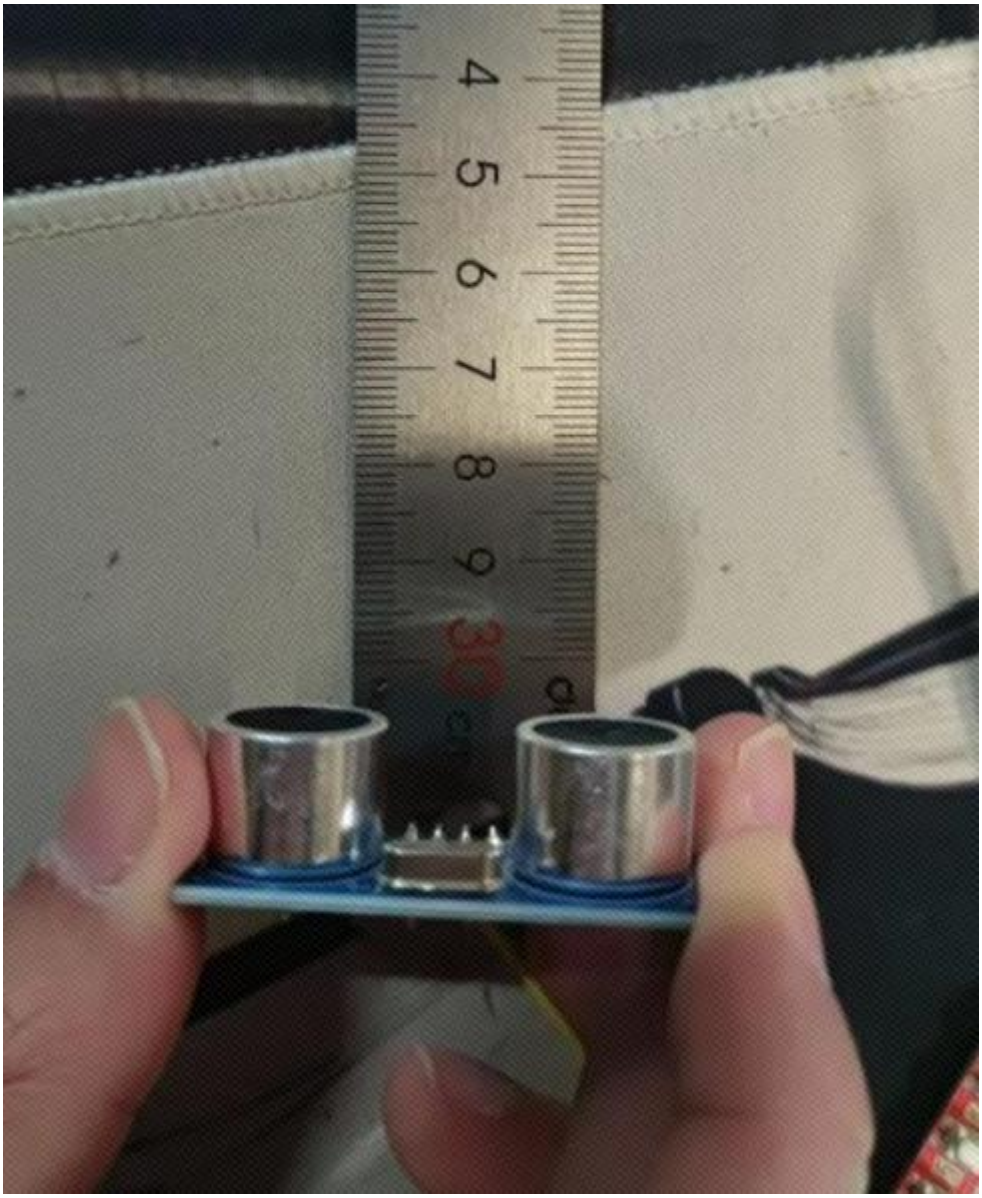

接下来在超声波测距模块 30cm 处放置一物体（需要大一些），将触发模式设置为 data_vld 信号高电平触发，按下开发板按键 k0，发现计算出的距离值为 30，微秒计数器的值为 1769，符合我们的预期。如下图所示：



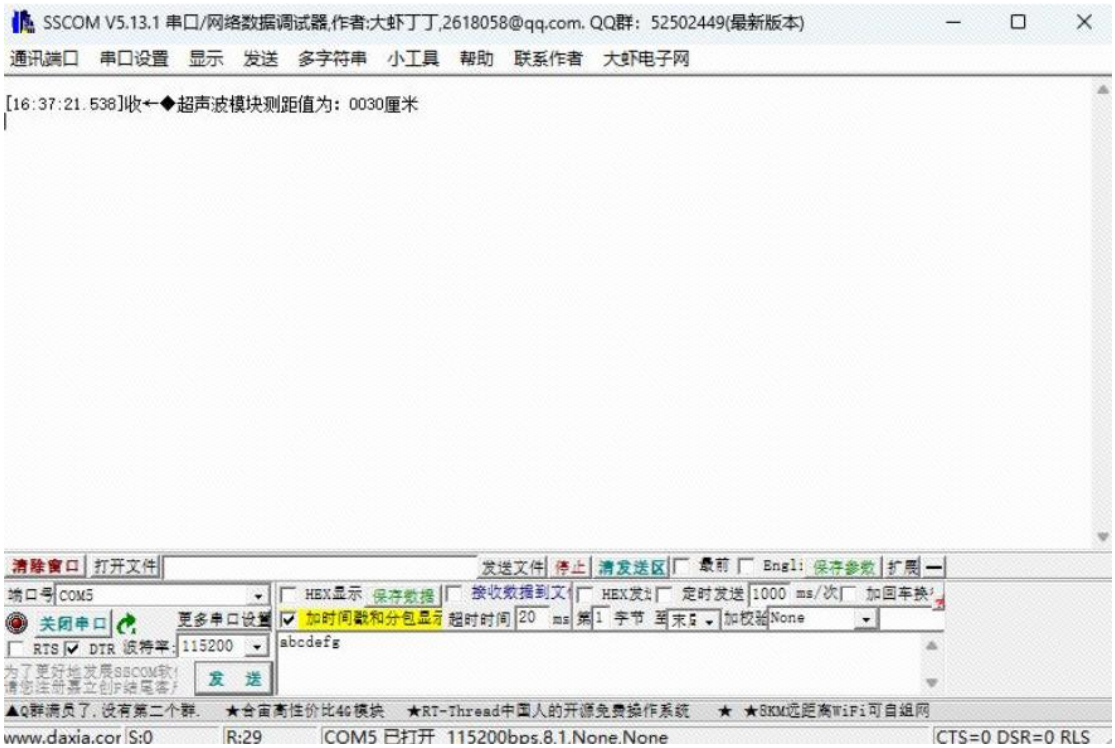

## 23.5. 实验现象

将板卡与超声波模块根据约束文件正确连接后，将程序下载进入板卡，按下板卡按键 k0，将会触发一次测距，并将得到的结果通过串口打印到上位机现象如下：

在 30cm 处放置一纸箱，按下板卡 key0 之后上位机接收如下：


4 5 6 7 8 9

串口打印信息如下：


SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com. QQ群：52502449(最新版本)
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者 大虾电子网
[16:37:21.538]收←◆超声波模块测距值为：0030厘米
清除窗口
打开文件
发送文件
停止
清发送区
端口号 COM5
关闭串口
更多串口设置
RTS DTR 波特率: 115200
发 送
abcdefg
▲Q群满员了，没有第二个群。 ★合宙高性价比4G模块 ★RT-Thread中国人的开源免费操作系统 ★ ★8KM远距离WiFi可自组网
www.daxia.cor S:0 R:29 COM5 已打开 115200bps,8,1,None,None CTS=0 DSR=0 RLS