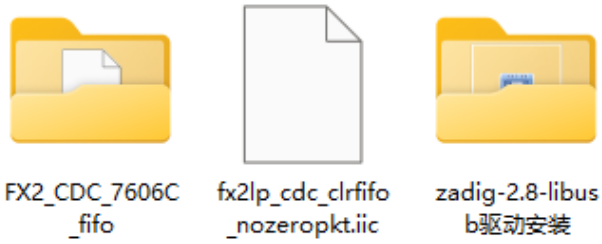
1 基于 FX2 的 7606C 的 8 通道 USB CDC 串口采集 FIFO 存储设计

工程源码	|--60k_FX2_CDC_7606C_fifo |--138k_FX2_CDC_7606C_fifo |-- FX2_CDC_7606C_fifo
相关视频课程	本教程无视频课程
本实验支持小梅哥 Xilinx AC720&高云 ACG720 开发板，如果您手头的硬件不支持本实验，您可以学习本实验的理论内容，也可以跳过本节内容，继续后续内容的学习。	

文档所涉及文件说明

本次实验配套的文件如下图，各文件功能如下所示：



- FX2_CDC_7606C_fifo：7606C 的 8 通道 USB CDC 串口采集工程源码
- fx2lp_cdc_clrfifo_nozeropkt.iic：需要烧录的 USB 固件，将 USB 配置为串行设备
- zadig-2.8-libusb 驱动安装：安装 libusb 需要的驱动程序。

章节导读

前面的章节讲解了 USB CDC 的原理与实现方法。本章主要介绍基于 ACG720 开发板 FX2 芯片，使用 USB CDC 协议，通过 SlaveFIFO 接口实现 USB CDC 虚拟串口下发指令，FPGA 将收到的的数据转化成控制命令对 ACM7606C 的采样频率、数据采样个数以及采样通道进行合理配置，采集完成后的数据经 FIFO 缓存后，通过串口传输到电脑的功能。同时，系统可解析波特率、数据位等串口参数，并利用数码管显示，方便调试与观察。

1.1 USB CDC

关于 USB CDC 的原理与实现方法，在前面的章节已介绍。具体链接如下：

【开源】使用 CY7C68013 实现 USB CDC 串口功能，含 68013 固件和 FPGA 测试程序

https://www.corecourse.cn/forum.php?mod=viewthread&tid=30141

(出处: 芯路恒电子技术论坛)

1.2 ACM7606C 模块简介

ACM7606C 数据采集模块使用的是 ADI 公司的 16 位 8 通道同步采样模数转换器 AD7606C，模块图如下图所示。

图 1-1 AD7606C 模块图

AD7606C 是一款具有八通道的 16 位同时采样模拟到数字数据采集系统（DAS）。每个通道包含模拟输入箝位保护、可编程增益放大器（PGA）、低通滤波器（LPF）和 16 位逐次逼近寄存器（SAR）模数转换器（ADC）。还包含灵活的数字滤波器、低漂移 2.5 V 精密参考电压源、驱动 ADC 的参考缓冲器以及灵活的并行和串行接口。

同时所有通道均能以高达 1MSPS 的吞吐速率采样。输入箝位保护电路可以耐受最高达±21V 的电压。无论以何种采样频率工作，其模拟输入阻抗均为 1MΩ。这是一个固定的输入阻抗，不随 AD7606C 采样频率而变化。这种高模拟输入阻抗消除了在 AD7606C 前面的驱动器放大器的需要，允许直接连接到源或传感器。因此，双极电源可以从信号链上移除。

芯片对外提供 SPI 和并行的数字接口。当 AD7606C 的 8 个通道全部以 1MSPS 的最高速率进行转换时，数据输出速率达到 128Mbps，需要使用高性能

MCU 的 SPI 外设才能勉强该速率要求。因此可以使用 16 位并口来进行数据的传输，提高数据传输速率。当 AD7606C 应用在 FPGA 系统中的时候，使用 SPI 串行接口和并行接口都能够轻松的满足数据传输的速率需求。当在 FPGA 系统上应用 AD7606C 时，可以通过在 FPGA 上设计 AD7606C 控制转换逻辑，将转换结果数据直接存储到片上的存储器如 FIFO 或者 RAM 中，也可以存储到 FPGA 片外的存储器如 SRAM 或 SDRAM 中，然后由其他主控芯片如 MCU 或 DSP 读出，或者直接在 FPGA 内部进行数据的运算和处理。当然，由于 FPGA 片上可以设计软核控制器，也可以直接使用软核控制器完成数据的处理和传输工作。

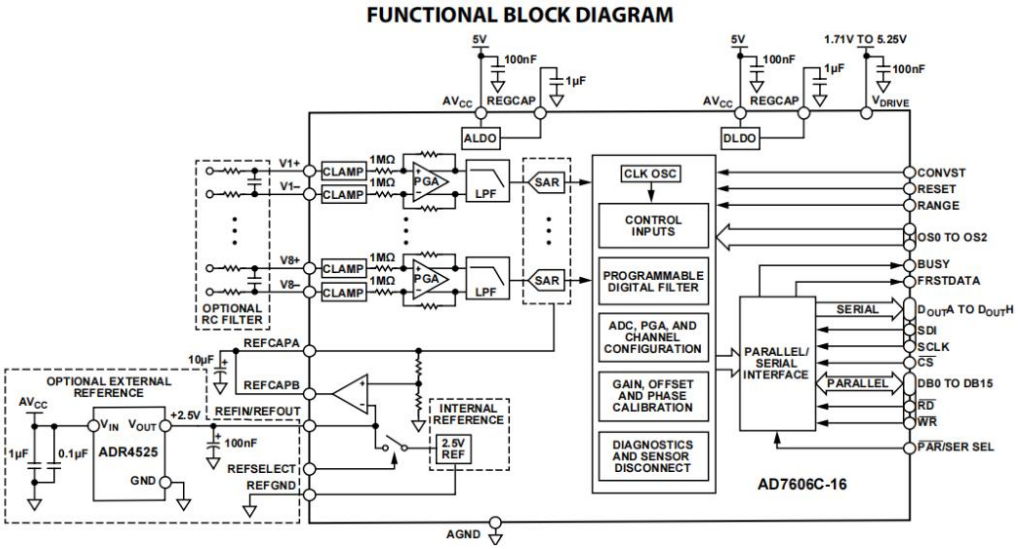


图 1-2 AD7606C 功能框图

AD7606C 根据采样方式不同具有多种驱动时序，本次实验采用的为软件模式下的并行输出，所以会涉及到 AD7606C 寄存器的配置。时序图由三部分组成：写寄存器时序、完成 AD 转换和读取 AD 数据。时序图如下所示。

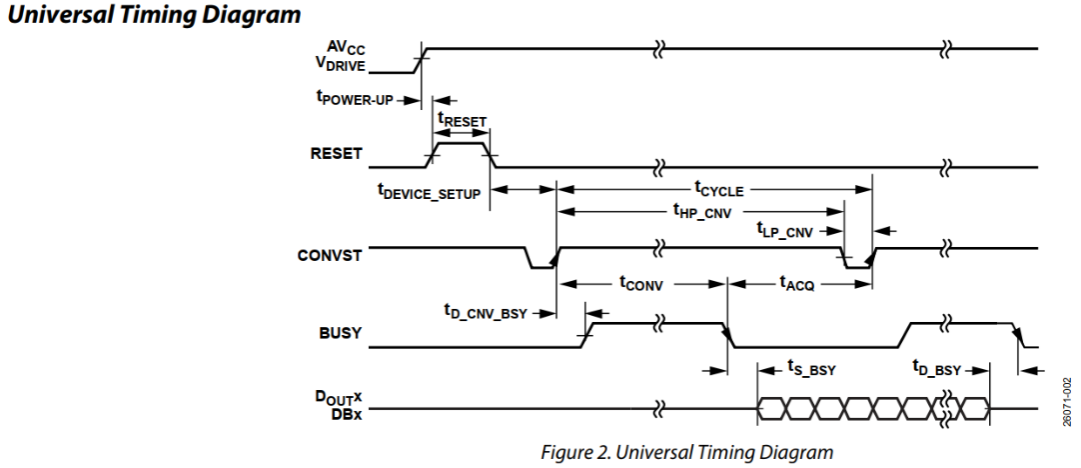


图 1-3 通用时序

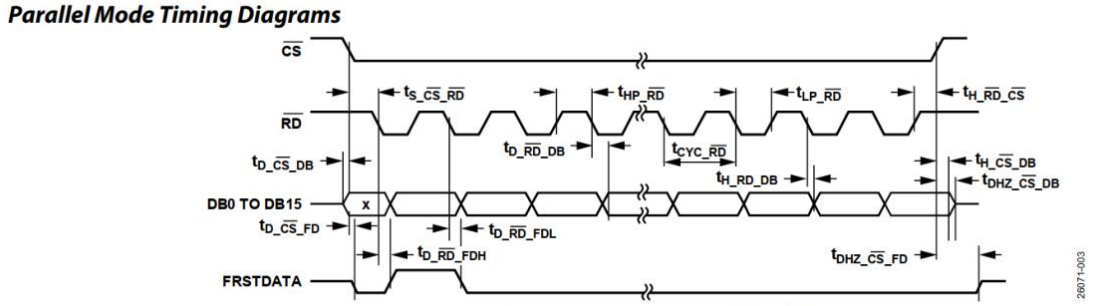


Figure 3. Parallel Mode Read, Separate $\overline{CS}$ and $\overline{RD}$ Pulses

图 1-4 并行时序，独立的 CS 和 RD 脉冲

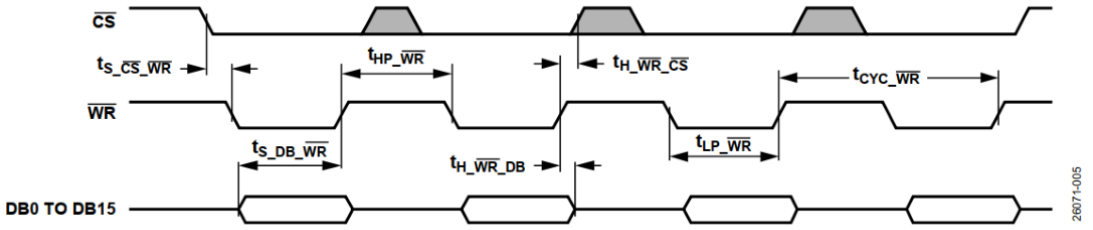


Figure 5. Parallel Mode Write Operation

图 1-5 并行模式写寄存器时序

当 CONVST 变为上升沿时，BUSY 信号转变为高电平，代表转换开始，知道 BUSY 的下降沿到来，代表数据已经转换完成，正在锁存至输出数据寄存器中，当 CS 变为下降沿时，数据将会被输送到总线上。并行模式下，每次数据的输出为 16 位，对应一个通道。

在理解本次设计原理后，接下来讲解本次实验的工程设计与验证。

1.3 系统整体设计

在 PC 端，USB CDC 驱动把 FX2 设备识别为一个 标准串口 (COMx)，对用户而言和普通 UART 串口无异。通过上位机发送指令数据，CDC 驱动将上位机应用层的数据打包成 USB Bulk 传输包，通过 USB 总线发送给 FX2 芯片。FX2 的端点 EP2 会把数据发送给 FPGA。FPGA 内部接收到指令，会将其解析为采样频率、数据采样个数以及采样通道。7606C 采集到的数据安装命令数据进行处理后存入 FIFO。当满足读条件后从 FIFO 中取出数据，FX2 将 FPGA 的数据装入 IN 端点 EP6 缓冲区，打包成 USB Bulk 数据上传给 PC，CDC 驱动在 PC 端将 USB 数据流还原成串口数据流。最终，通过上位机应用显示采集到的波形数据。

系统整体设计结构框图如下图 1-6 所示。

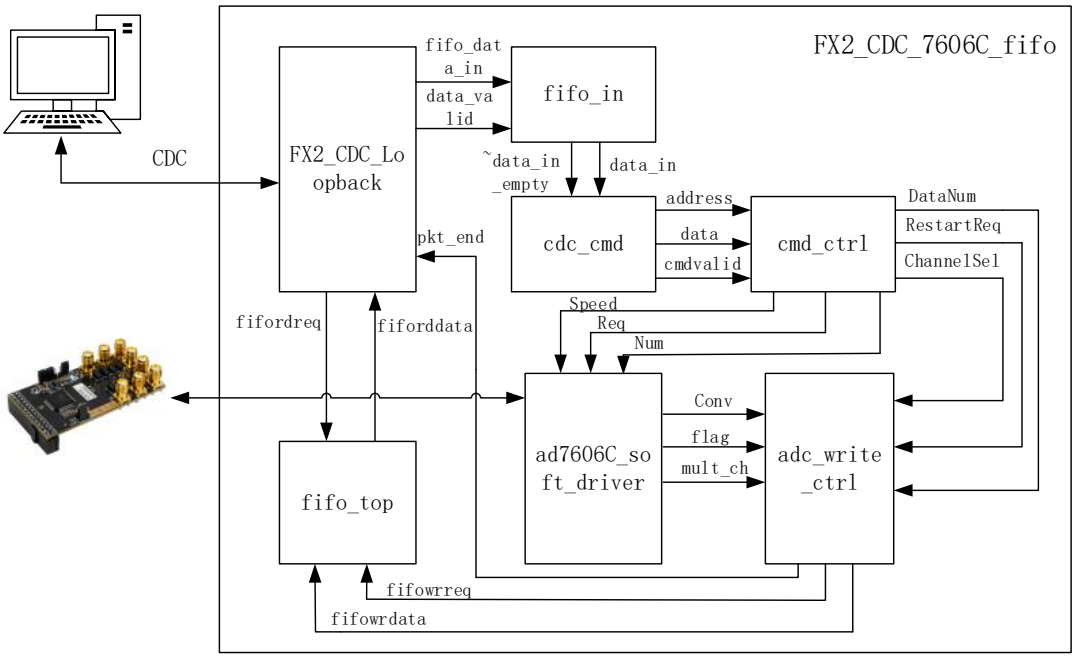


图 1-6 系统整体结构图

FX2_CDC_7606C_fifo：主体工程模块，整合 FX2 SlaveFIFO 接口、CDC 虚拟串口、7606c 驱动、接收转命令、命令转控制等模块。

FX2_CDC_Loopback：FX2 控制信号处理与数据传输模块，控制 FX2 的读写信号与数据的发送与接收，同时通过 SPI 读取上位机的设置数据。

SPI_Slave：通用从机模块。

ad7606C_soft_driver：7606C 驱动模块模块，对 AD7606C 型 8 通道 16 位 ADC 的数据转换控制并输出。

cdc_cmd：接收转命令模块，将 CDC 传输的指令数据帧进行拆解，得到需要的指令数据传送给别的模块进行处理。

cmd_ctrl：指令转控制模块，接收转命令模块接收到的数据转换为相应的控制数据。

adc_write_ctrl：状态控制模块，根据指令转控制模块的数据，控制写请求与写数据。

hc595_driver：数码管驱动模块，用于驱动 74HC595，发出数据和选通信号。

hex8：数码管动态扫描驱动模块，用于输出需要显示的数据、段选和位选值。

1.3.1 FX2_CDC_7606C_fifo 模块

FX2_CDC_7606C_fifo 模块，主要为整合各个模块以及其信号连接。

在 FX2_CDC_7606C_fifo 中，由于 7606C 的驱动时钟为 100M，而 FX2 的接口时钟信号为 96M，因此需要对部分信号进行跨时钟域处理。

对于 CDC 串口接收的指令数据，可以使用高云的异步 FIFO 进行处理，使用 FIFO 的非空信号作为读使能。例化代码如下：

```
    fifo_in fifo_in(
    .Data(fifo_data_in), //input [7:0] Data
    .Reset(~rst_n), //input Reset
    .WrClk(Clk_fx2), //input WrClk
    .RdClk(clk_100m), //input RdClk
    .WrEn(data_valid), //input WrEn
    .RdEn(~data_in_empty), //input RdEn
    .Q(data_in), //output [7:0] Q
    .Empty(data_in_empty), //output Empty
    .Full(data_in_full) //output Full
);
```

对于包结束信号 pkt_end，为持续 5 个 100M 周期的信号，我们寄存两拍处理即可，这样能解决亚稳态，不会丢信号。设计代码如下：

```
reg pkt_end_sync1, pkt_end_sync2;
always @(posedge fx2_ifclk or negedge rst_n) begin
if (!rst_n) begin
   pkt_end_sync1 <= 0;
   pkt_end_sync2 <= 0;
end else begin
   pkt_end_sync1 <= pkt_end;
   pkt_end_sync2 <= pkt_end_sync1;
end
end

wire pkt_end_96m = pkt_end_sync2;
```

当使用小梅哥上位机时，由于上位机处理数据顺序为低位在前，高位在后，因此例化的 FIFO 写数据端口需要高低 8 位进行互换。设计代码如下：

```
.Data({fifowrdata[7:0],fifowrdata[15:8]}), //input [15:0] Data
```

1.3.2 FX2_CDC_Loopback 模块

FX2_CDC_Loopback 模块用于 FX2 CDC 接收指令与发送数据以及 FX2 控制信号的控制，并通过 SPI 协议，使用 SPI_Slave 模块读取当前串口上位机的主要寄存器的工作状态。

当上位机通过 CDC 串口发送采样指令时，FX2 的端点 2 的空标志 fx2_flagb 会拉高（低有效），此时代表 FD 数据线上有数据。因此当 fx2_flagb 为 1 且写指令 FIFO 未满时，拉低读控制信号 fx2_slrd（低有效）和输出使能信号 fx2_sloe（低有效）并将 FD 总线的数据通过 fifo_data_in 存入写指令 FIFO。设计代码如下：

```
// 接收状态机（从 FX2 读取数据）
reg [1:0] rx_state;
localparam RX_IDLE = 0,
           RX_READ = 1,
           RX_WRITE = 2;

always @(posedge fx2_ifclk or negedge reset_n) begin
  if (!reset_n) begin
    rx_state <= RX_IDLE;
    fifo_data_in <= 8'd0;
    data_valid <= 0;
  end else begin
    case (rx_state)
      RX_IDLE: begin
         data_valid <= 0;
        if (fx2_flagb == 1) begin  // 有数据可读
          rx_state <= RX_READ;
        end
      end
      RX_READ: begin
         fifo_data_in <= fx2_fdata;
         data_valid <= 1;
         if(fx2_flagb == 0)
            rx_state <= RX_WRITE;
         else
            rx_state <= RX_READ;
         end
```

```
      RX_WRITE: begin
         data_valid <= 0;
         rx_state <= RX_IDLE;
         end
    endcase
  end
end

    //读控制信号
  always @(*) begin
  if((fx2_flagb == 1'b1)&&(~fifo_full))begin
     slrd_n = 0;
     sloe_n = 0;
  end else begin
     slrd_n = 1;
     sloe_n = 1;
  end
  end
```

当 fx2_flagc 为高且 FIFO 非空，即 FX2 端点 6 FIFO 未写满且 FPGA FIFO 内部已经有采集的数据时，拉高读请求以及写控制信号 fx2_slwr（低有效）。当 fx2_flagc 拉低时，即 FX2 端点 6 FIFO 已经写满，此时需立即拉低写请求以及拉低 fx2_slwr。设计代码如下：

```
  always @(posedge fx2_ifclk or negedge reset_n) begin
  if (!reset_n) begin
    tx_state <= TX_IDLE;
    slwr_n <= 1;
    fifordreq_r <= 0;
  end else begin
    case (tx_state)
      TX_IDLE: begin
         if (!fifo_empty && fx2_flagc_r) begin
            tx_state <= TX_REQ;
         end
      end
      TX_REQ: begin
            fifordreq_r <= 1;
            slwr_n <= 0;  // 写数据
         if(fifo_empty || !fx2_flagc)begin
```

```
            tx_state <= TX_WRITE;
            fifordreq_r <= 0;
            slwr_n <= 1;
         end
         else
            tx_state <= TX_REQ;
      end
      TX_WRITE: begin
         fifordreq_r <= 0;
         slwr_n <= 1;
         tx_state <= TX_IDLE;
      end
    endcase
  end
end

  assign fx2_slwr  = fx2_flagc ? slwr_n : 1'b1;
  assign fifordreq = fx2_flagc ? fifordreq_r : 1'b0;

  always @(*) begin
  if (slwr_n == 1'b1) data_out = 8'dz;
  else data_out = fifo_data_out;
  end
```

fx2_faddr 用于选择四个 FIFO 中的哪一个与 FD 总线相连，因此需要控制其读写状态需要使用哪个 FIFO。设计代码如下：

```
always @(*) begin
   if ((rx_state != RX_IDLE)|| faddr_hold)
      faddr = 2'b00;
   else if (tx_state != TX_IDLE)
      faddr = 2'b10;
   else
      faddr = 2'b10;
end
```

具体波形如下：

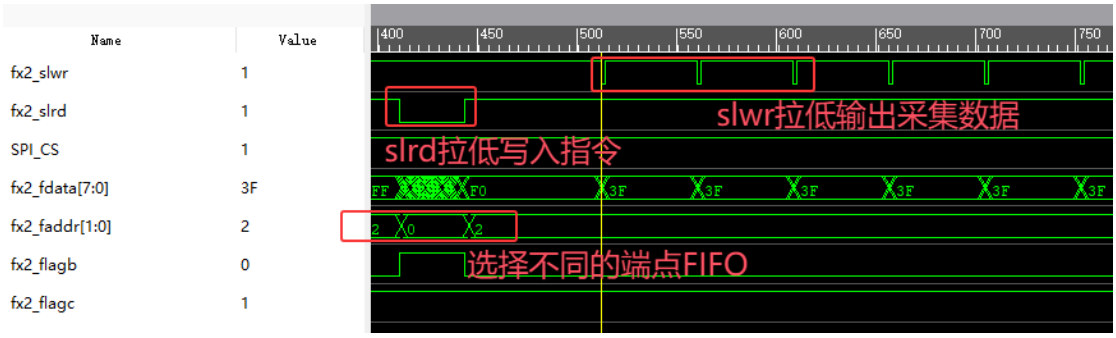


图 1-7 7606C 采集数据

fx2_pkt_end 为最后一包结束标志，当采集完指定数量的数据且 FIFO 读空后，延迟一段时间后拉低 fx2_pkt_end 并保持一段时间。设计代码如下：

```
always @(posedge fx2_ifclk or negedge reset_n) begin
if (!reset_n) begin
    delay_cnt   <= 0;
    delay_start <= 0;
end
else begin
    if (pkt_end) begin
        delay_start <= 1; // 触发开始延时
    end

    if (delay_start && fifo_empty) begin
        if (delay_cnt < 8192) begin
            delay_cnt <= delay_cnt + 1 ;
        end
        else begin
            delay_cnt   <= 0;;
            delay_start <= 0; // 延时完成，清除标志
        end
    end
    else begin
        delay_cnt <= 0;
    end
end
end

assign fx2_pkt_end = ((delay_cnt >= 8185) && (delay_cnt <= 8191)) ? 1'b0 : 1'b1;
```

使用串口助手时，SPI_Slave 模块为通用 SPI 从机接口，通过 FX2 的端点 0

解析串口上位机的指令寄存器，将 FPGA 作为 SPI Slave，使用 SPI 协议读取当前串口上位机的主要寄存器的工作状态。当串口上位机连接 CDC 虚拟串口后，通过 SPI 从 USB 读取串口上位机的配置寄存器。

对于 CDC 串口：

Byte 0~3: 波特率（Baud Rate），DWORD，LSB First

Byte 4: 停止位数量（Stop Bits）:

0 = 1 位，1 = 1.5 位，2 = 2 位

Byte 5: 校验（Parity）:

0 = None，1 = Odd，2 = Even，3 = Mark，4 = Space

Byte 6: 数据位（Data Bits），通常为 8

1.3.3 cdc_cmd 模块

cdc_cmd 模块为接收转命令模块，将串口传输过来的指令数据帧进行拆解，得到需要的指令数据传送给别的模块进行处理。

串口口一次发送的命令数据内容为 40 个字节，为了实现通过串口修改这些寄存器的值，需要对发送一次的数据进行拆解才能实现，对于设计的数据帧，一帧数据一共 8 个字节，包含帧头、帧尾、地址段、数据段。帧格式如下表所示：

表 1 帧格式说明表

数据	D0	D1	D2	D3	D4	D5	D6	D7
功能	帧头 0	帧头 1	地址 addr	data[31:24]	data[23:16]	data[15:8]	data[7:0]	帧尾
值	0x55	0xA5	xx	xx	xx	xx	xx	0xF0

从上表中可以看出，每帧数据一共 8 个字节，分别用 D0~D7 表示，其中，D0 和 D1 两个数据作为帧头，其值固定为 0x55、0xA5，D7 作为帧尾，其值固定为 0xF0。帧头和帧尾的作用是为了准确识别数据帧，确保接收的数据是我们需要分析的。D2 代表的是要操作的寄存器地址，D3 为要写入寄存器的数据的 24~31 位，D4 为要写入寄存器的数据的 16~24 位，D5 为要写入寄存器的数据的 8~15 位，D6 为要写入寄存器的数据的 0~7 位。

该模块的作用就是将串口接收到的数据拆解成上述帧格式，将 D2 作为地址 address 输出，指定修改哪个寄存器，D3~D6 共 32 位作为数据 data 输出，控制 AD7606C 进行相应的配置。下面将对模块中的部分代码进行说明：

首先是得到帧命令数据，每有一个 rx_done 信号，便将接收到串口的数据做移位存储，连续存储 8 个字节的数据就得到一帧命令数据。代码如下所示：

```
always@(posedge Clk)
 if(rx_done)begin
    data_str[7] <= rx_data;
    data_str[6] <= data_str[7];
    data_str[5] <= data_str[6];
    data_str[4] <= data_str[5];
    data_str[3] <= data_str[4];
    data_str[2] <= data_str[3];
    data_str[1] <= data_str[2];
    data_str[0] <= data_str[1];
 end
```

最后是判断得到的帧命令数据是否正确，当数据符合 D0 为 8'h55，D1 为 8'hA5，D7 为 8'hF0，则代表该数据格式正确，会生成一个指令正确信号 cmdvalid 输出到指令转控制模块，并将数据进行输出，代码如下所示：

```
always@(posedge Clk or negedge Reset_n)
 if(!Reset_n) begin
    address <= 0;
    data <= 0;
    cmdvalid <= 0;
 end else if(r_rx_done)begin
    if((data_str[0] == 8'h55) && (data_str[1] == 8'hA5) && (data_str[7] ==
8'hF0))begin
       data[7:0] <= data_str[6];
       data[15:8] <= data_str[5];
       data[23:16] <= data_str[4];
       data[31:24] <= data_str[3];
       address <= data_str[2];
       cmdvalid <= 1;
    end
 end
 else
    cmdvalid <= 0;
```

1.3.4 cmd_ctrl 模块

指令转控制模块 cmd_ctrl 将从接收转命令模块接收到的数据转换为相应的控制数据，首先将对寄存器进行说明，其功能和地址分别如下表所示：

表 2 寄存器说明表

名称	地址	位宽	功能简介
RestartReq	0	1	重新开始采集请求寄存器，向该寄存器写入任意值即可启动新一轮的采样存储传输
ChannelSel	1	8	通道设置寄存器，共 8 位， 对应了 8 个通道的数据存储开关，如果某通道对应的设置为 1，则该通道的采样结果就会被存入 FIFO 并通过串口发送，注意对应的 2 进制的位，不是 10 进制，比如设置通道 3，对应 01 地址需要写入的值为 04(100)，而不是 03(011)。

DataNum	2	32	数据个数寄存器，设定总共采集传输多少个数据。注意，该寄存器设置的是总共采集的数据个数， 假设设置采集 100 个数据，ChannelSel 为 0000_0011b， 则实际每个通道采样的次数就是 50，2 个通道的数据加起来是 100 个。假设设置采集 100 个数据，而且设置了 ChannelSel 为 0011_0011b，则实际每个通道采样的次数就是 25，4 个通道加起来采集 100 个数据
ADC_Speed_Set	3	25	ADC 采样速率设置寄存器。该寄存器用来设置 ADC 每多久执行一次转换。 由于 ADC 的最大采样速率为 1Msps，所以可以通过设置该寄存器的值来让 ADC 的采样速率在 1~1Msps 范围内调整，以适应不同的应用场景。ADC_Speed_Set=1000000000/10/speed-1，其中 speed 就是实际要设置的采样速率。
ADC_soft_modle	4	32	AD7606C 软件模式配置寄存器，该寄存器用来配置 AD7606C 在软件模式下每个寄存器的参数。

根据表 2 中的内容，地址 cmd_addr 为 0 时，产生 RestartReq；cmd_addr 为 1 时，得到通道设置数据 cmd_data[7:0]；cmd_addr 为 2 时，得到需要采样的数量 cmd_data[31:0]；cmd_addr 为 3 时，得到设置的采样速率的值；cmd_addr 为 4 时，得到需要配置的寄存器以及需要配置的值，代码如下所示：

```
always@(posedge Clk or negedge Reset_n)
if(!Reset_n)begin
    ChannelSel <= 8'b1111_1111;
    DataNum <= 32'd32;
    ADC_Speed_Set <= 32'd9999;
    RestartReq <= 1'b0;
    ADC_soft_modle <= 32'd0;
end
else if(cmdvalid)begin
   case(cmd_addr)
       0: RestartReq <= 1'b1;
       1: ChannelSel <= cmd_data[7:0];
       2: DataNum <= cmd_data[31:0];
       3: ADC_Speed_Set <= cmd_data[24:0];
       4: ADC_soft_modle <= cmd_data;
     default:;
  endcase
end
else
    RestartReq <= 1'b0;
```

1.3.5 adc_write_ctrl 模块

adc_write_ctrl 模块用于控制写 FIFO 的请求与结束信号。

当接收到开始采样请求信号后，需等待 Conv 信号为高时才能拉高采样使能，Conv 在 AD7606C 数据转换时为低电平，转换完成后拉高。在 Conv 为高时拉高

采样使能，可以防止接收的采样请求信号正好在数据转换期间，造成数据错位。设计代码如下。

```
//采样控制逻辑，每次采样请求信号到来开始采样，采样个数满了停止采样。
   always@(posedge Clk or negedge Reset_n)
   if(!Reset_n)
      sample_en_reg <=  1'b0;
   else if(RestartReq)
      sample_en_reg <=  1'b1;
   else if(data_cnt >= DataNum )
      sample_en_reg <=  1'b0;

   always@(posedge Clk or negedge Reset_n)
   if(!Reset_n)
      sample_en <=  1'b0;
   else if(sample_en_reg && Conv )
      sample_en <= 1'b1;
   else if(data_cnt >= DataNum )
      sample_en <=  1'b0;
```

data_cnt 是采样计数信号，用于控制采样使能的状态。设计代码如下。

```
//采样个数计数器，在采样使能阶段，每个 flag 到来的时候计数器自加 1
   always@(posedge Clk or negedge Reset_n)begin
   if(!Reset_n)
      data_cnt <=  32'd0;
   else if(sample_en)begin
      if((adc_data_flag & ChannelSel) && (!fifowrfull) )
         data_cnt <=  data_cnt + 1'd1;
      else
         data_cnt <=  data_cnt;
   end
   else
      data_cnt <=  32'd0;
   end
```

写请求信号由 flag，通道选择、使能信号共同控制。当采样使能为高，且通道选择和 adc_data_flag 相同时，拉高写请求，写入对应当的通道数据。当通道选择为 0 时，在上位机使用通道 1 传输测试数据。设计代码如下。

```
//用于仿真或产生测试数据
   reg [15:0]adc_test_data;
```

```
//测试数据，当 sample_en 为 1 时，每个周期使 adc_test_data 加 1
always@(posedge Clk)
adc_test_data <= sample_en ? (adc_test_data + 1'b1) : 16'd0;

always@(posedge Clk)begin
if(ChannelSel == 8'b0)
   fifowrreq <= ((adc_data_flag[0]) &&  sample_en && (!fifowrfull));
else
   fifowrreq <=  ((adc_data_flag & ChannelSel)  && sample_en) &&(!fifowrfull) ;
end

always@(posedge Clk)
   fifowrdata <= (ChannelSel == 8'b0)?adc_test_data: adc_data_mult_ch;
```

具体波形如下：

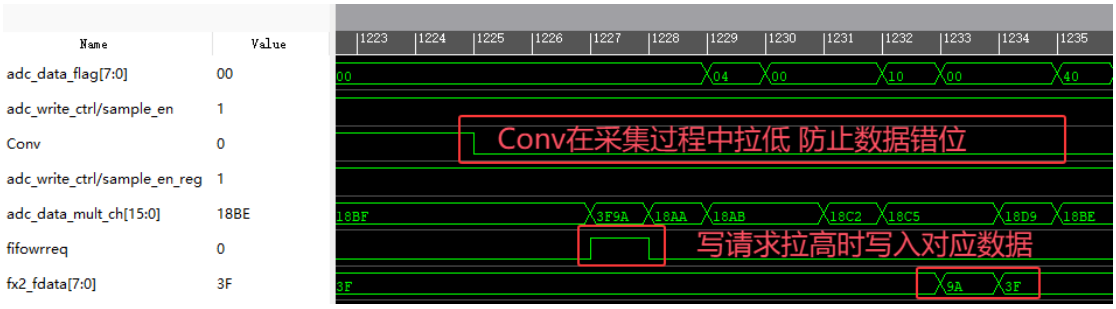


图 1-8 FIFO 写数据

1.3.6 ad7606C_soft_driver 模块

AD7606C 控制器驱动模块 ad7606c_soft_driver，该控制器实现了对 AD7606C 型 8 通道 16 位 ADC 的数据转换控制并输出。使用该控制器时，用户无需关心 AD7606C 的具体控制时序，一切都在控制器内部完成，用户只需要像使用并行 ADC 一样取用数据即可。

该控制器在工作时会根据主机的指令对采样频率进行修改，当信号转换完成后便对 ADC 写控制器发出 data_flag 信号，控制器将转换完成对应通道的数据写进 FIFO。

其他模块的详细介绍可参考开发板对应学习资料。

【高云】【ACG720】高云 ACG720 138k&60k 教学开发板产品使用自助服务手册

https://www.corecourse.cn/forum.php?mod=viewthread&tid=29765

(出处: 芯路恒电子技术论坛)

主要模块已经介绍完了，完整的顶层文件代码请自行查看例程文件，接下

来进行板级验证。

1.4 板级验证

本次实验使用 ACG720-60K 以及其板载 CY7C68013 以及 ACM7606C 模块进行验证。

1.4.1 系统所需硬件

1. ACG720-60k 开发板
2. 电源线一根
3. Type-c 数据线一根
4. ACM7606C 模块一个
5. 下载器一个

1.4.2 硬件连接

本次设计硬件连接如下图所示：

本次实验需要使用一根 Type-c 连接开发板右上方的 USB 接口与电脑，7606C 模块对齐开发板 GPIO1 脚插入。

连接好下载器与电源适配器。

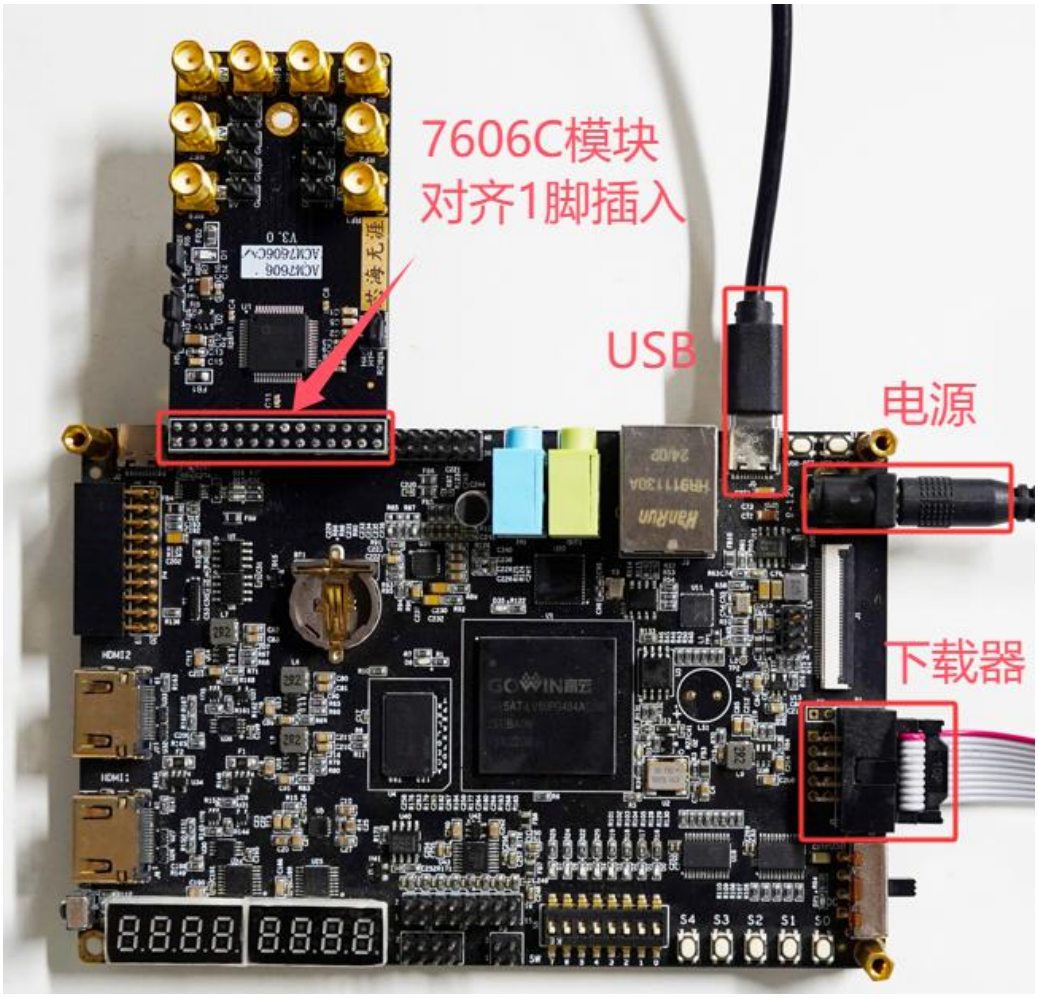


图 1-9 硬件连接

连接完毕后接下来即可进行 FX2 固件的烧写。

1.4.3 烧写 FX2 固件

如果想把 USB 配置为 CDC 虚拟串口设备，需要通过固件里的 设备描述符 + 配置描述符 + 接口描述符 来配置为 CDC 串口，当固件把自己枚举为 CDC-ACM 类设备时，操作系统就会加载内置驱动，把它映射成一个虚拟串口。

我们在资料中提供了一个 fx2lp_cdc_clrfifo_nozeropkt.iic 固件。

打开 CyControlexe 软件，使用 USB 线连接开发板 USB 接口和电脑的 USB 接口，软件会显示出一个名为 Cypress FX2LP No EEPROM Device 的设备，选中该设备，点击 Program -> FX2 -> RAM，或选择 EEPROM 掉电不丢失，如下所图 1-10 所示。

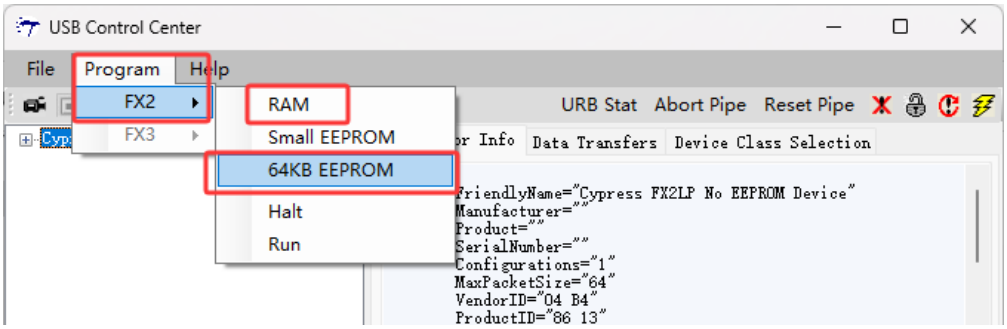


图 1-10 固件烧写界面

在弹出的文件选择框中选择 fx2lp_cdc_clrfifo_nozeropkt.iic 固件，然后点击打开，软件即开始下载固件到 FX2 芯片的 RAM/EEPROM 中，如下图 1-11 所示。

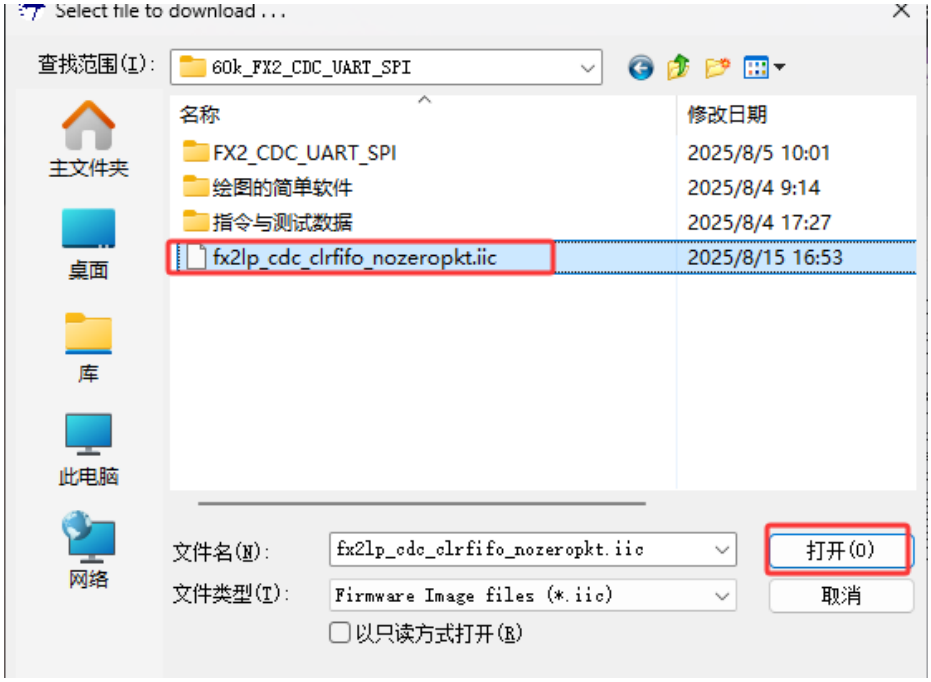


图 1-11 烧写固件

程序下载完成后，软件左下角会显示 Programming Succeeded，提示程序下载完成，按下开发板 USB 旁的 USB-RST 键，接下来 FX2 芯片会重新枚举，然后在 Control Center 软件中可以看到识别不到设备，这是因为 USB 已经映射为了虚拟串口，可以在设备管理器中看到 USB 被识别为了串行设备，如图 1-12 所示。

图 1-12 USB 被识别为串行设备

1.4.4 管脚分配

经过上述工作，所有代码都已经设计完毕，硬件环境也已搭建完成。 接下来进行管脚分配，如下表所示：

表 3 管脚分配表

	信号名	引脚号	信号名	引脚号
基本管脚	clk	Y18	SW	G22
	reset_n	F15		
数码管	st_cp	C2	sh_cp	F4
	ds	M18		
FX2	fx2_slwr	L20	fx2_slrd	K19
	fx2_sloe	J15	fx2_slcs	J16
	fx2_pkt_end	J14	fx2_ifclk	L19
	fx2_flagc	J17	fx2_flagb	H18
	fx2_faddr[0]	K13	fx2_faddr[1]	H13
	fx2_fdata[0]	G18	fx2_fdata[1]	G17
	fx2_fdata[2]	G20	fx2_fdata[3]	J20
	fx2_fdata[4]	G13	fx2_fdata[5]	G16
	fx2_fdata[6]	K14	fx2_fdata[7]	G15
	SPI_SCLK	L15	SPI_MOSI	K16
	SPI_MISO	L16	SPI_CS	L14
AD7606C	ad7606_reset_o	A19	ad7606_rd_n_o	A18
	ad7606_cs_n_o	F14	ad7606_convst_o	A15
	ad7606_busy_i	F13	ad7606_os_o[0]	A14
	ad7606_os_o[1]	A13	ad7606_os_o[2]	A16
	ad7606_db_i[0]	C13	ad7606_db_i[1]	B13
	ad7606_db_i[2]	D14	ad7606_db_i[3]	D15
	ad7606_db_i[4]	C14	ad7606_db_i[5]	C15
	ad7606_db_i[6]	B15	ad7606_db_i[7]	B16

	ad7606_db_i[8]	D17	ad7606_db_i[9]	C17
	ad7606_db_i[10]	E16	ad7606_db_i[11]	D16
	ad7606_db_i[12]	B17	ad7606_db_i[13]	B18
	ad7606_db_i[14]	C18	ad7606_db_i[15]	C19

1.4.5 下载与验证

程序编译成功后即可以进行下载验证。

点击 GOWIN 软件的 Programmer 标志。检测到下载器后点击 Save 确认，如图 1-13 所示。

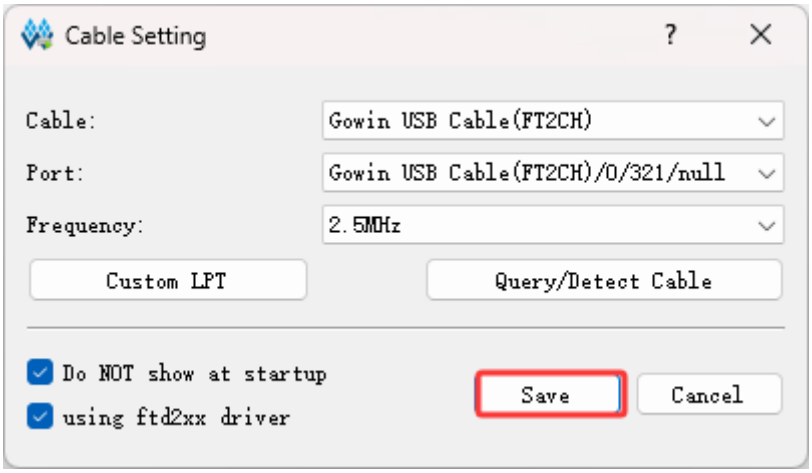


图 1-13 识别下载器

之后点击 进行程序烧录，下方的输出栏出现如下图 1-14 所示的提示即代表烧录成功。

```
Info    Target Cable: Gowin USB Cable(FT2CH)/0/321/null@2.5MHz
Info    Target Device: GW5AT-60B(0x0001481B)
Info    Operation "SRAM Program" for device#1...
Info    Frequency Updated: "15MHz"
Info    User Code is: 0x00001DC3
Info    Status Code is: 0x70026020
Info    Finished.
Info    Cost 4.4 second(s)
```

图 1-14 烧录成功

1.4.5.1 串口调试助手通信

在前面接收转命令模块中介绍到数据帧格式对 AD7606C 的五个寄存器进行

配置。

例如，PC 端要设置采样数据个数(DataNum 寄存器，地址为 2)为 16384(0x4000)个，发送数据帧内容：0x55 0xA5 0x02 0x00 0x00 0x40 0x00 0xF0。

PC 端要设置采样速率(ADC_Speed_Set 寄存器，地址为 3)为 1M，则对应的 ADC_Speed_Set 值为 1000000000/10/1000000 - 1 = 99（0x00000063），则发送的数据帧内容为：0x55 0xA5 0x03 0x00 0x00 0x00 0x63 0xF0。

当上述设置都设置完成后，就可以向 0 号寄存器写入任意值，来开始一次采样传输了。数据帧内容可以为 0x55 0xA5 0x00 0x00 0x00 0x00 0x00 0xF0，这里需要注意的是 0 号寄存器必须放在最后，因为 0 号寄存器负责启动 ADC，ADC 在未配置完全的情况下开始启动，数据很容易输出错误值。

开始传输数据帧命令发送完成之后，AD7606C 就能实现以 1M 的采样速率，对 1 个通道进行采样（本次实验以通道 1 为例），共采集 32768 个数据。五个寄存器对应的配置如表 4 所示：

表 4 AD7606C 数据帧格式配置表

寄存器名称	数据帧数据
ChannelSel	55 A5 01 00 00 00 01 F0
DataNum	55 A5 02 00 00 80 00 F0
ADC_Speed_Set	55 A5 03 00 00 00 63 F0
ADC_soft_modle	55 A5 04 00 00 00 00 F0
RestartReq	55 A5 00 00 00 00 00 F0

接下来打开串口调试助手，端口选择串行设备（具体端口号以自己电脑为准）并运行。如图 1-15 所示。

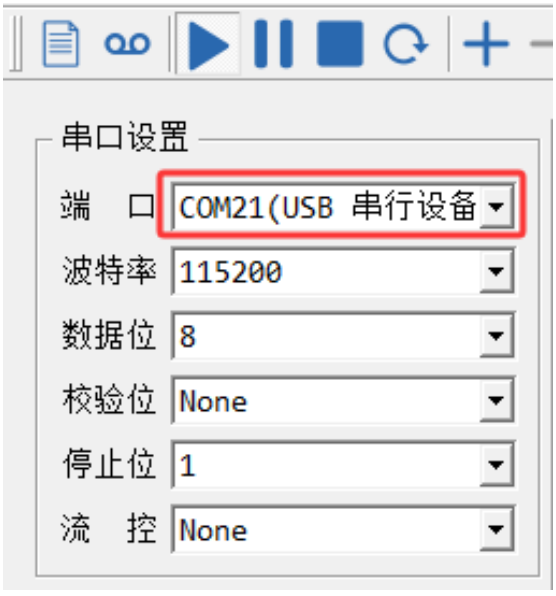


图 1-15 串口助手设置

在串口助手发送的数据格式如下：（1 通道、采 16384 字节、采样速率 1M）

55A50100000001F0 55A50200004000F0 55A50300000063F0 55A50400000000F0 55A50000000000F0

发送完成后可以看到串口调试助手在不断的接收数据。如图 1-16 所示。从图中可以看出一共接收到 16384 个数据，这是因为设置的 ADC 采样数量为 16384>>1，ADC 采样数据是 16 位的，串口是以字节（8 位）为单位进行发送的，所以通过串口接收到字节数应该是 16384 个数据，这也就是说明接收到的数据的个数没错。

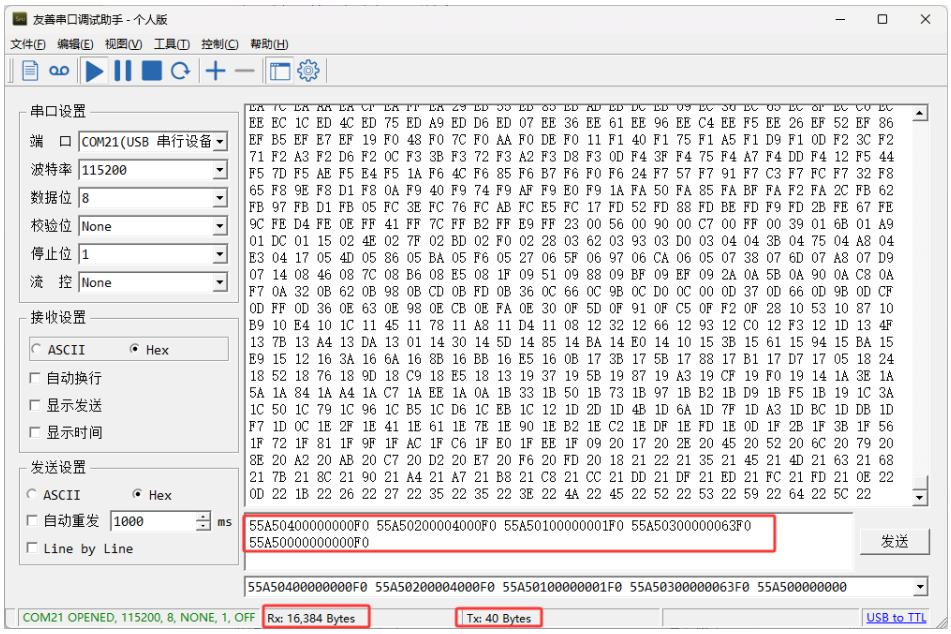


图 1-16 接收数据

此时修改串口助手的串口设置，在开发板板载数码管上可以看到显示出对应的 16 进制波特率。将拨码开关 **SW0** 拨至上方，可以看到数码管上显示出对应的数据位、校验位和停止位的信息。（波特率 9600、数据位 5、校验位 Even、停止位 1）如下图所示：

图 1-17 数码管显示

1.4.5.2 libusb 驱动安装

对于 Windows 自带的 CDC 串口驱动来说，libusb 驱动更加稳定，可以直接使用 bulk 传输，速率比 CDC 串口快，适用于大批量高速采样数据的传输。接下

来使用 libusb 驱动进行设计的验证。

双击打开配套案例中提供的 libusb 驱动安装工具 zadig-2.8.exe ，在弹出的 Zadig 界面中点击 Options，点击 List All Devices 列出所有驱动。如下图所示。

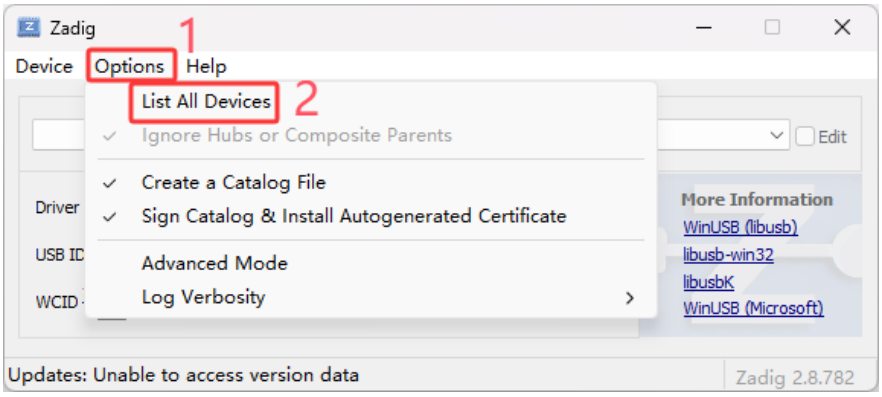


图 1-18 列出所有驱动

接着在驱动选择栏选择 USB Serial 驱动，之后点击 Replace Driver 替换 CDC 驱动。如下图所示。

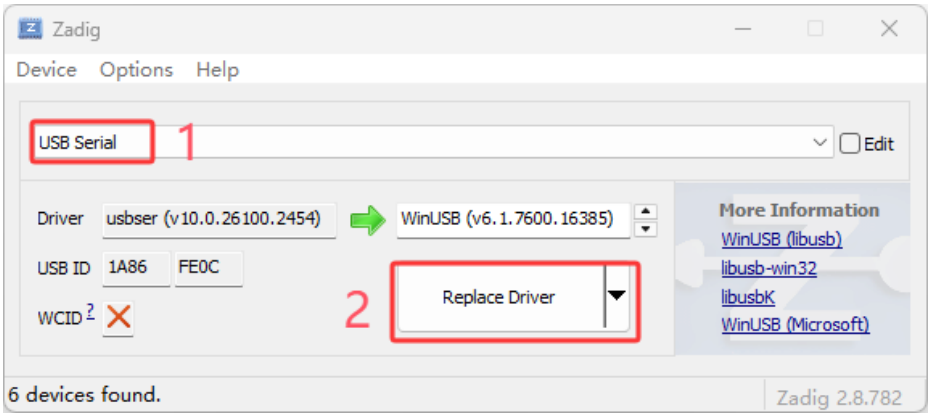


图 1-19 替换驱动

libusb 安装较慢，若安装失败，可以重启电脑再试，安装成功如下图所示。

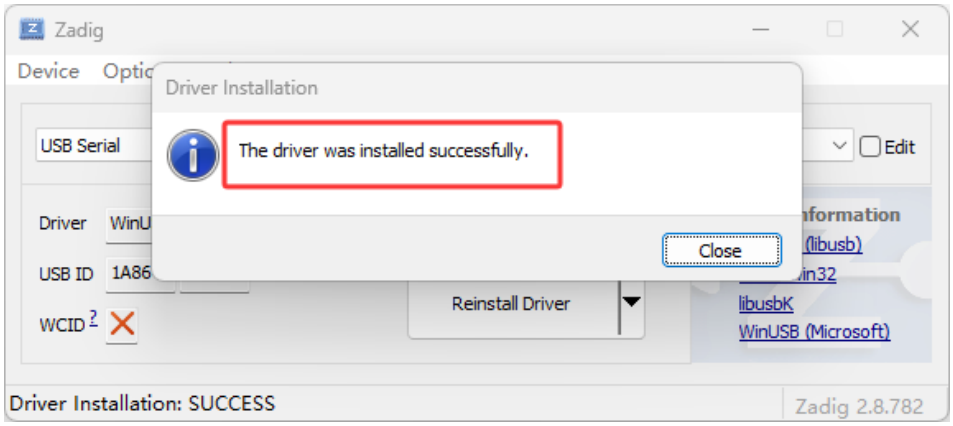


图 1-20 安装成功

安装成功后，设备管理器端口中的串行设备会消失，在通用串行总线设备下会出现 USB Serial 端口。如下图所示。

图 1-21 设备管理器

当不使用时 libusb 或需要使用 FX2 时，可直接在设备管理器卸载 USB Serial 驱动即可。

1.4.5.3 数据采集上位机通信

安装 libusb 驱动后

前面通过串口调试助手采集数据时，每次保存数据都需要重新点击“接收保存文件”一栏，修改寄存器参数的时候，都需要重新计算，然后发送命令，修改之后也不能直接实时观察到数据波形，使用起来不是很方便。基于上述问题，我们设计了上位机软件“小梅哥数据采集仪”进行数据采集，上位机内部直接对命令进行了构建，用户只需要在界面上对采样参数进行设置，便可以实时观测到数据变化，软件获取位置如下：

基于 QT 的小梅哥数据采集系统上位机软件使用说明

https://www.corecourse.cn/forum.php?mod=viewthread&tid=29728

(出处: 芯路恒电子技术论坛)

双击 QSCOPE_xxx.exe 打开上位机软件，软件初始界面如下。

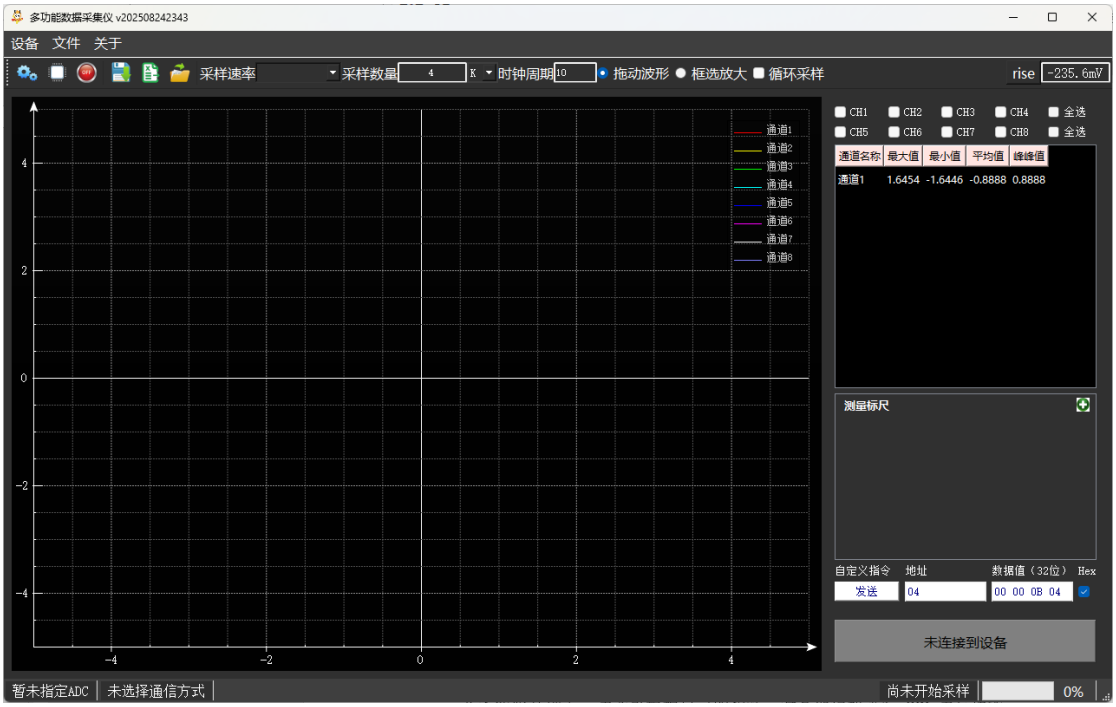


图 1-22 上位机初始界面

本次实验使用该软件的操作方法如下：

1. 选择通信方式，选择 LibUSB，点击检索设备识别出 USB Serial 后点击 OK，如下图所示：

图 1-23 设置通信方式

2. 选择 ADC 型号，点击选择 ACM7606C 后点击 OK，如下图所示：

图 1-24 选择 ADC

3. 接着点击打开设备按钮，连接设备，如下图所示：

4. 最后 AD7606C 连接好信号发生器，设置需要的采样速率、采样数量，因为 AD7606C 使用的是 100M 时钟，因此时钟周期设为 10（单位 ns）。接着选择需要采集的通道，点击右下方开始采样。如下图所示：

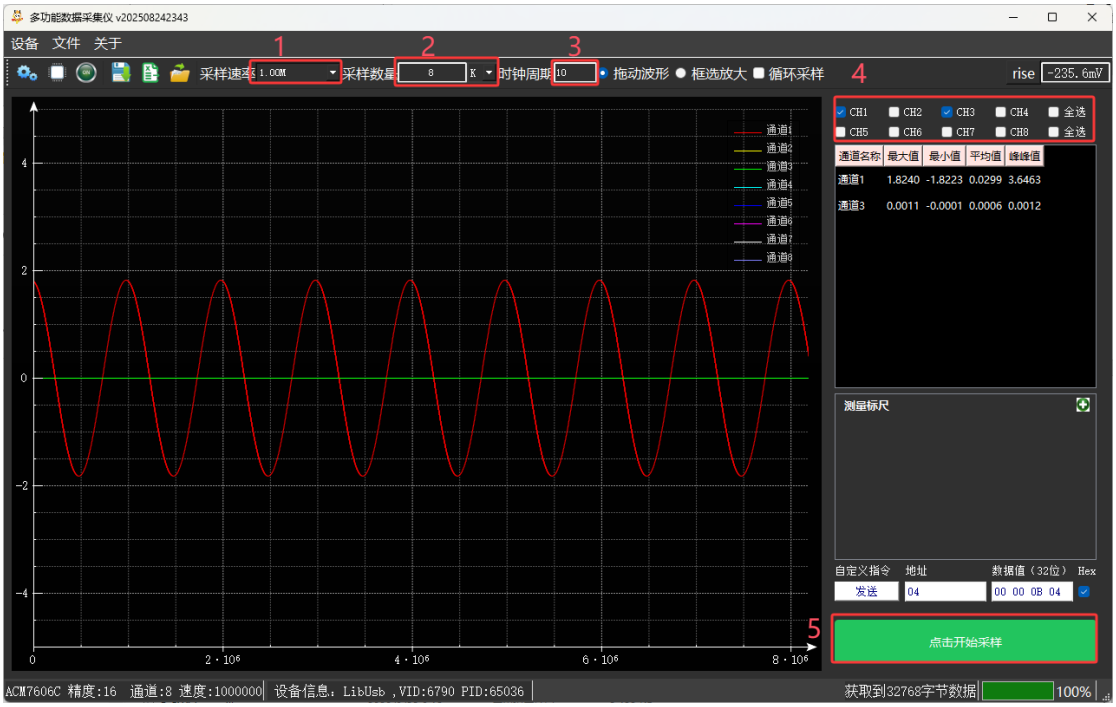


图 1-25 开始采样

可以看到正确的采集出了波形，采样数量为 8K*2 通道*2=32768 字节。

软件还可以计算波形的频率，点击右侧测量标尺旁的加号，在出现的标尺中点击白色的 A1，鼠标移动到波形上时会出现标尺，此时在需要测量波形合适的位置点击鼠标左键，即可固定标尺，用同样的步骤操作 A2，即可测量两个标尺之间的时间为 1000000ns（1S/1000000ns=1000Hz）符合信号发生器提供的频率。如下图所示：

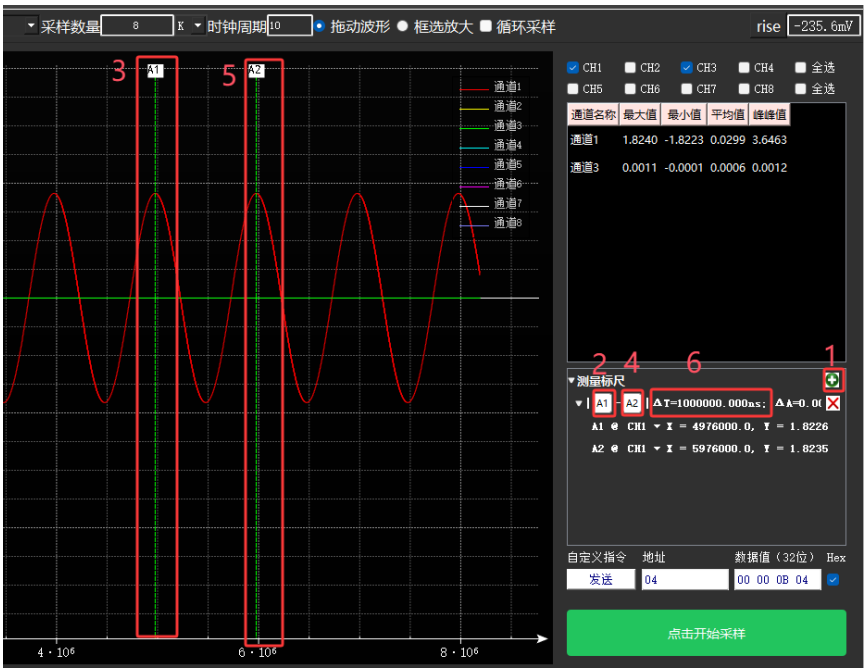


图 1-26 频率测量

注 1：由于 CDC 速率限制，在较高采样率采集时，FX2 的端点 6 FIFO 未及时读取，会造成 FLAGC 拉高，采集的数据因此会产出错位。

注 2：当不使用 USB CDC 串口或需要使用正常 USB 功能时，按下 USB 接口旁的 USB-PGM 的同时按一下 USB-RST 即可。

1.5 总结与思考

本节实验基于 FX2 芯片实现了 USB 转 CDC 虚拟串口功能，并通过 FPGA 实现了通过 CDC 虚拟串口发送 AD7606C 采样指令以及采样数据接收的功能。本实验介绍了了 USB CDC 协议的基本框架、枚举过程与数据传输机制，以及 FX2 SlaveFIFO 接口的读写时序控制方法和 ADC128S102 的工作原理。建议读者能够跟随本实验内容，完整的进行整个实验。

在本实验基础上，读者可根据自己的想法，对其他模块进行 SPI 读写功能的设计。