# 1 基于 AC6103 的 ACM9238 数据采集 DDR 缓存串口发送实验

| | |
|---|---|
| 工程源码 | \|--AC6103_ACM9238_DDR2_UART.rar |
| 相关视频课程 | 暂无无相关视频课程 |

章节导读

本节内容将介绍基于 ACM9238 模块利用串口进行数据采集的相关内容。通过串口调试助手发送相关指令，从而实现对 ACM9238 模块的采样频率、数据采样个数以及采样通道的合理配置，采集完成后的数据通过 DDR2 进行缓存，然后通过串口再传输到电脑。用户可以在电脑上通过串口调试工具进行指令的下发，并以文件的形式保存接收到的数据，然后使用 MATLAB 软件进行进一步的数据处理分析。

## 1.1 背景介绍

在计算机广泛应用的今天，数据采集系统在许多领域都有着十分重要的应用。

数据采集是计算机与外部物理世界连接的桥梁，通过数据采集工作，自然界的许多模拟量信息能够借助计算机进行保存，分析，还原等操作。技术实践中，我们只需要制订上位机(PC)与移动数据采集器的通信协议，就可以实现两 者之间阻塞式通信交互过程。

数据采集系统往往由传感器、模拟多路开关、放大器，采样保持器、 AD 转换器、计算机及外设等组成。在农业、工业、日常生活和航空航天等领域，尤其是在对信息实时性能要求较高或者恶劣的自然环境中，数据采集有其应用的必要性。比如说：在工业生产和科学技术研究的各行业中，常常有利用 PC 或工控机配合末端传感器对诸如液位、温度、压力、频率等参数进行实时监控和记录的需求，这些环境往往有时候并不适合人类直接作业，或者即使人类进行直接作业，也无法达到和计算机自动采集分析处理某一个任务的实施效果。再比如说：在航天航空领域，卫星数据采集系统利用航天遥测、遥控、遥监等技术，对航天器远

地点进行各种监测，并根据需求进行自动采集，经过卫星传输到数据中心处理后，送给用户使用。

谈到数据采集，不得不说说数据转换器。现在常用的数据转换方式是通过 数据采集板卡进行，常用的有如 A/D 卡以及 422 、485 等总线板卡，我们今天要进行的实验，就是利用 ACM9238 数据采集卡实施数据采集。

## 1.2 工程目标任务

本工程介绍如何通过利用 AC6103 上的 DDR2 资源 ，实现 ACM9238 模块的串口数据采集功能。

本工程在实验条件下，期望达到如下功能要求：

1、实验通过数据采集模块实现模数转换，传递给开发板。在这里，我们采用 ACM9238 双通道数据采集模块作为数据采集卡进行数据采集。为了真实模拟数据采集的实验环境，我们借助信号发生器作为信号源，使用时可以通过设置其不同频率进行采样观察效果。

2、使用相关的串口通信软件，通过串口发码指令，可以设定需要采集的字节数，选择采集通道号，启动采集，或者直接使用我们的数据采集上位机进行数据采集。

3、使用相关的串口通信软件，可以按设定的采集参数，接收采集的数据。数据通过串口发送到串口调试软件。将读取到的数据我们进行 DAT 文件的保存，便于后期分析。

4、采集到的数据经过 MATLAB 波形分析波形，能够得到和输出波形一致的输出，无数据丢失，无杂波。后期可结合实际情况，增加加噪声评定的环节。

## 1.3 ACM9238 模块简介

该模块如下图 1-1 所示。ACM9238 模块配合前端模拟信号调理电路，实现了±5V 电压范围内信号的高速采样。该模块共使用 2 路完全相同的 AD 采样和信号调理电路，构成了双通道高速 AD 采样电路。两路 ADC 电路完全独立，结构和元器件参数相同，确保了两个通道有较高的一致性。本模块与 FPGA 连接采用并行接口，每路 ADC 包括 12 位数据信号（ADC_DATA），1 位时钟信号（ADC_CLK），1 位超量程指示信号（ADC_OTR），该模块接口图如下图 1-2 所示。

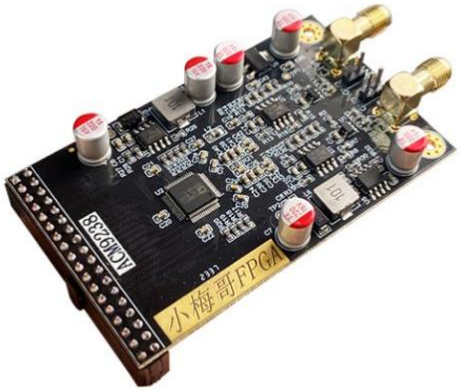


图 1-1 ACM9238 模块图

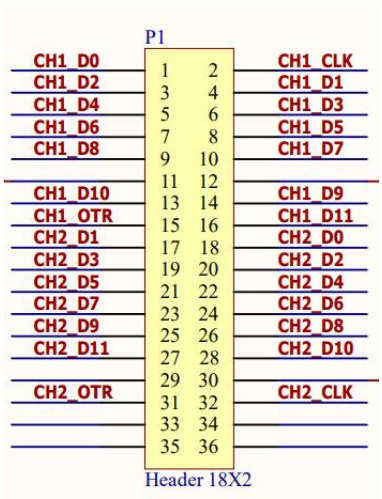


图 1-2 ACM9238 模块接口图

使用该模块时，仅需 FPGA 为每路 ADC 提供一路时钟信号，ADC 则会在每个时钟周期输出一个 12 位的采样结果。当 9238BSTZ 模拟输入端接-5V 至+5V 之间变化的正弦波电压信号时，其转换后的数据也是成正弦波波形变化，转换波形如下图 1-3 所示，从图中可以看出 9238BSTZ 采集到的数据是无符号数据。

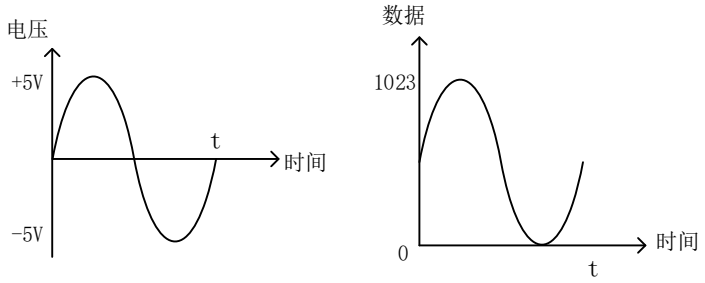


图 1-3 9238BSTZ 正弦波模拟电压值（左）、数据（右）

本模块采样率上限为 50Msps，采样率就等于 FPGA 提供给 ADC 的时钟频率。如需使用低于时钟频率的采样率，可以依旧给 ADC 提供 50MHz 的时钟信号，但在 FPGA 内部，对 50Msps 的采样结果数据进行抽取重采样的方法实现。比如期望以 1Msps 的采样速率采样，则只需要每间隔 50 个采样数据取一个结果存储或使用，其他 49 个数据直接舍弃，这样就能实现 1MSPS 的采样率了。十分不建议采用直接对提供给 ADC 芯片的时钟信号降频以实现降低采样率的效果的方法，因为时钟太低，会影响 ADC 芯片内部采样保持电路的工作情况，导致采样误差偏大。

本模块可用于小梅哥全系列 FPGA、SOC、Zynq 开发板，包括国产开发板和各核心板的评估底板。AC620、AC6103、ACX720、ACZ7015、ACZ702、智多晶 FPGA 开发板（AC208-SA5Z）、AC608 评估底板、AC601 评估底板、AC675 评估底板。

## 1.4 程序设计

整体来说，程序的硬件架构如下所示，PC 端发送指令信息到 FPGA，FPGA 对 ADC 采集的数据进行读取缓存，通过串口发送到 PC 端。

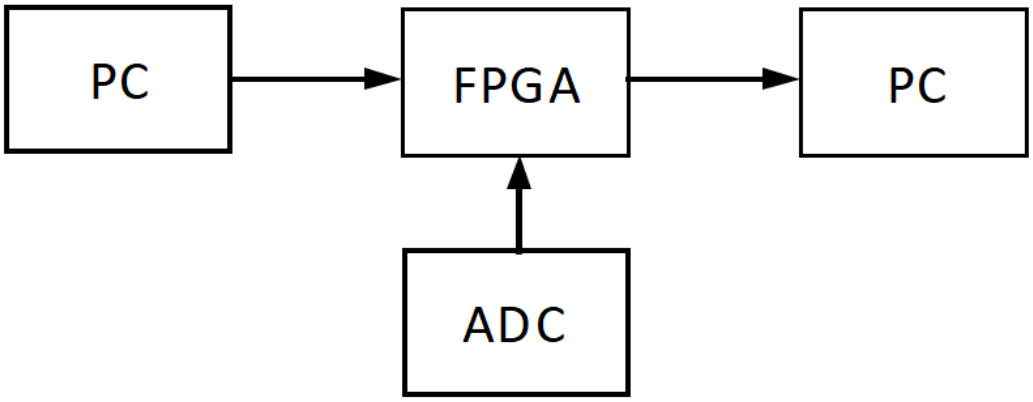


图 1-4 硬件架构图

下面给出简易的状态转移图。

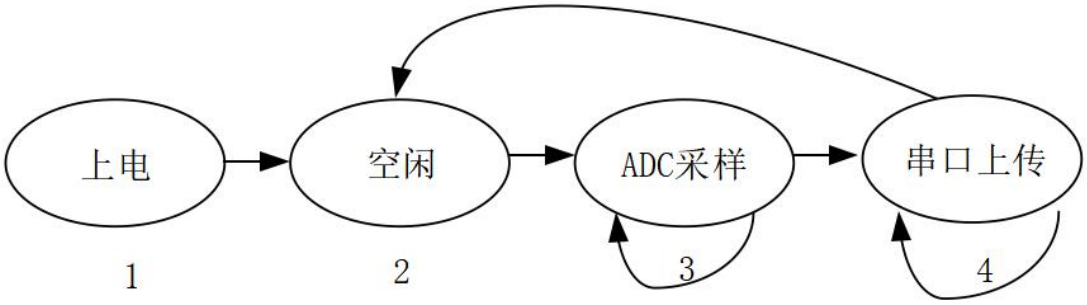


图 1-5 状态转移图

初步分析程序的状态机核心部分，主要分为上电，空闲，采样，串口上传四

个部分。

程序上电后，进入状态 2 空闲状态，此时可以通过串口指令设置采样个数，设置采样频率，下发采样开始的指令。

FPGA 收到采样开始指令后，进入状态 3 开始进行 ADC 采样。采样的同时数据直接储存在 DDR2 中。如果未达到采样个数，则进行状态 3 的循环，如果达到采样个数，则进入状态 4 串口上传。如果串口上传完成，则返回空闲状态，如果串口上传未完成，则继续进行串口上传。为实现本次实验功能，工程设计了 9 个子模块，如下图 1-6。

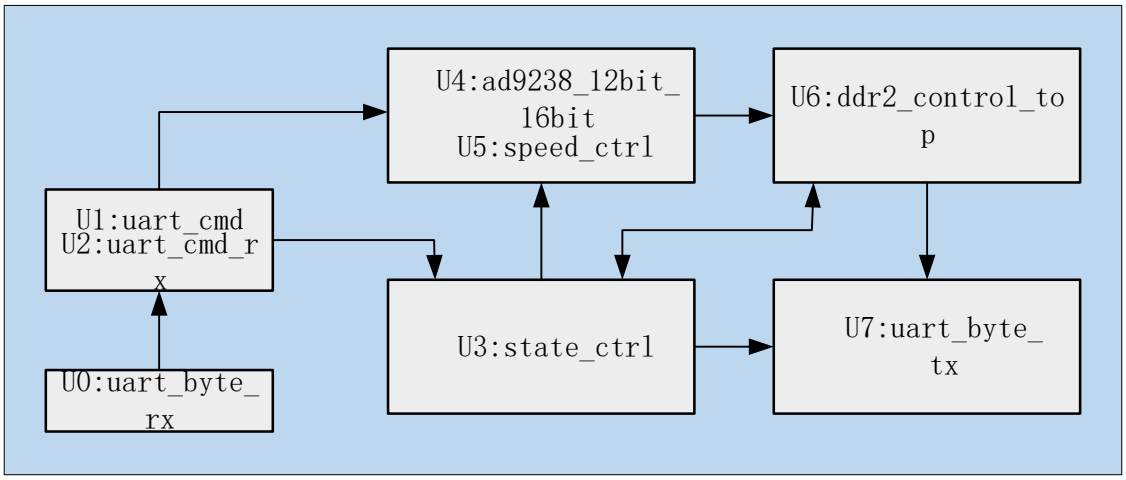


图 1-6 系统模块框图

内部各个子模块功能如下：

1、U0：串口接收模块的指令接收功能。

2、U1 、U2：接收到的指令进行翻译拆解，指令分类。

3、U3：状态机模块，协调各个模块的信号控制，程序状态的总控制模块。

4、U4、U5：ACM9238 数据输入 12bit 到 16bit 的转换、速度控制模块。

5、U6：DDR2 的含 fifo 封装模块，主要负责整个数据的存储功能。

6、U7：串口数据输出模块。

### 1.4.1 串口接收/串口发送模块

本次实验的指令配置信息由 PC 端从串口调试软件下发到 FPGA ，PFGA 接收来自串口的指令信息，设置采样的数据量、采样通道和采样频率，接收到的指令信息将由下一个模块进行解析。本次实验的波特率设置为 115200，图 1-7 串口接收模块为串口接收模块图。

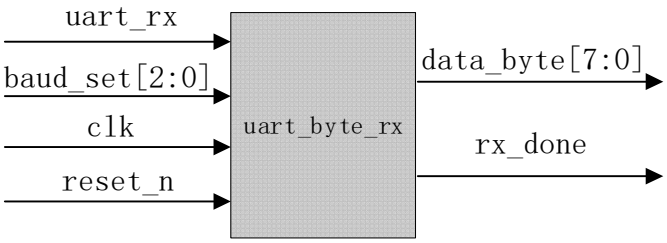


图 1-7 串口接收模块

表 1-1 串口接收模块接口功能描述

| 接口名称 | I/O | 功能描述 |
|---|---|---|
| clk | I | 输入工作时钟，频率为50MHz |
| reset_n | I | 模块复位，低电平复位 |
| uart_rx | I | 串口数据接收端口 |
| baud_set[2:0] | I | 波特率设置端口 |
| rx_done | O | 1byte 数据接收完成标志信号 |
| Data_byte[7:0] | O | 指令数据八位输出端口 |

本实验 ACM9238 采集到的数据缓存在 DDR2 中，通过 DDR2 读 fifo 来读出 DDR2 中的数据，串口发送模块把采集的数据发送到 PC 端，不过 DDR2 读 fifo 输出数据数据位宽为 16 位，需要在 state_ctrl 模块进行位宽转换，最终转换成 8 位位宽的数据。串口发送模块发出的数据可在 PC 端通过串口调试助手观察到，串口发送模块如图 1-8。

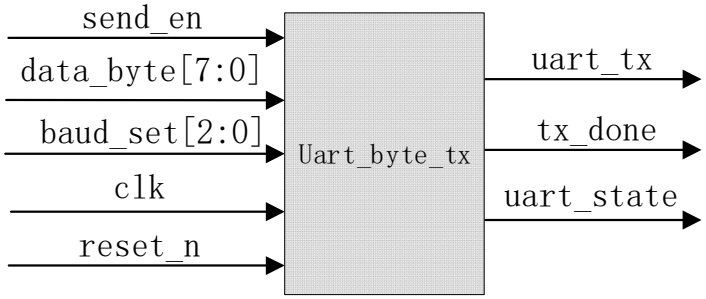


图 1-8 串口发送模块

该模块的信号说明如下表 1-2 所示。

表 1-2 串口发送模块接口功能描述

| 接口名称 | I/O | 功能描述 |
|---|---|---|
| clk | I | 输入工作时钟，频率为50MHz |
| reset_n | I | 模块复位，低电平复位 |
| send_en | I | 发送使能信号 |
| baud_set[2:0] | I | 波特率设置端口 |
| Data_byte[7:0] | I | 待传输 8bit 数据 |
| tx_done | O | 1byte 数据发送完成标志 |

| | | |
|---|---|---|
| uart_tx | O | 串口输出信号 |

### 1.4.2 指令接收/指令解析模块

这两个模块的作用，是把串口接收到的指令进行拆解和识别。从串口接收 到指令数据后，uart_byte_rx 模块将串口数据从串行信号转为 8 位的并行数据 uart_rx_data。指令数据在模块中根据包头与包尾信息判断 8 字节指令信息是否有效。指令有效，uart_cmd 会读取寄存器地址设置信息和采集数据的配置数据信息，串口指令接收模块如图 1-9。

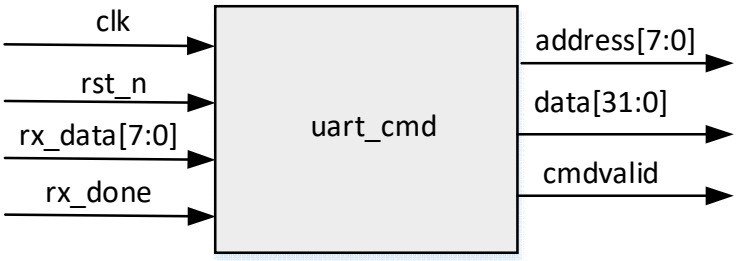


图 1-9 串口指令接收模块

表 1-3 串口指令接收模块接口功能描述

| 接口名称 | I/O | 功能描述 |
|---|---|---|
| clk | I | 输入工作时钟，频率为50MHz |
| rst_n | I | 模块复位，低电平复位 |
| rx_done | I | 1byte 数据接收完成标志信号 |
| rx_data[7:0] | I | 指令数据输入端 |
| cmdvalid | O | 指令信息解析完成标志信号 |
| address[7:0] | O | 寄存器地址信息数据 |
| data[31:0] | O | 采样设置配置数据信息 |

从 uart_cmd 模块接收到的 8 位并行数据在 uart_cmd 模块中被解析出地址、数据和使能信号。uart_cmd_rx 将前面解析出的数据，作为各个类型，分别储存在各个寄存器中，对采样的数据量和采样速率进行设置，uart_cmd_rx 如图 1-10。

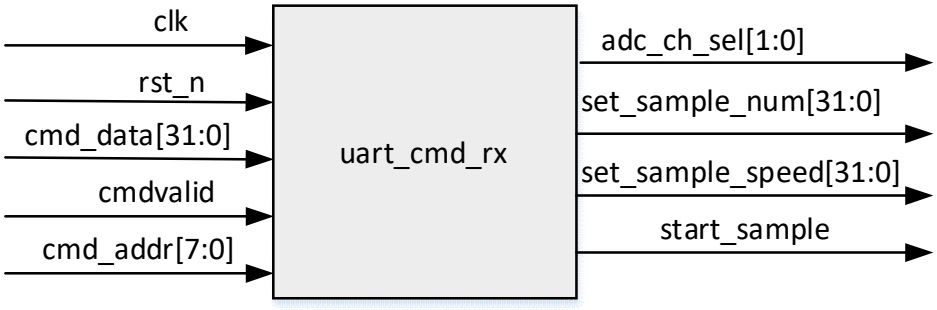


图 1-10 指令解析模块

表 1-4 指令解析模块接口功能描述

| 接口名称 | I/O | 功能描述 |
|---|---|---|
| clk | I | 输入工作时钟，频率为50MHz |
| rst_n | I | 模块复位，低电平复位 |
| cmdvalid | I | 指令信息解析完成标志信号 |
| Cmd_addr[7:0] | I | 寄存器地址信息数据 |
| cmd_data[31:0] | I | 采样设置配置数据信息 |
| start_sample | O | 采样使能信号 |
| adc_ch_sel[1:0] | O | 采样通道选择信号 |
| set_sample_num[31:0] | O | 采样数据量信号 |
| set_sample_speed[31:0] | O | 采样率设置信号 |

在这里，我们给出指令解析的代码和讲解。

```
always@(posedge clk or negedge rst_n)
if(!rst_n)begin
uart_baud_set <= 3'd4; //默认115200bps
adc_ch_sel <= 2'b00;
set_sample_num <= 16'd32768;
start_sample <= 1'b0;
end
else if(cmdvalid)begin
case(cmd_addr)
    0: start_sample <= 1'b1;
    1: adc_ch_sel <= cmd_data[1:0];
    2: set_sample_num <= cmd_data[15:0];
    4:
        begin
        adc_ch_sel <= cmd_data[1:0];
        set_sample_num <= cmd_data[23:8];
        start_sample <= 1'b1;
  end
    5: uart_baud_set <= cmd_data[2:0];
  default:;
endcase
 else
   start_sample <= 1'b0;
 end
```

ACM9238 的控制指令，由 8 个字节的数据组成，前两个字节 D0，D1 用 55 、A5，最后一个字节 D7 帧尾用 F0，标明这是一个接收的指令，第三个字节 D2，

标明的是控制存储地址，本工程中，我们定义 00 是发送启动命令，01 是采样通道号，02 是采样深度。

串口一次发送的数据内容为 1 个字节，为了实现通过串口修改这些寄存器的值，需要发送多个字节才能实现，为此，设计了简单的串口数据帧，该帧一帧数据共 8 个字节，包含帧头、帧尾、地址段（决定任务设定目标）、数据段。帧格式如下表 1-5 所示。

表 1-5 指令帧格式

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

下面讲解一下典型的参数设置方法：

如果采 512 字节的数据，则设置为：55 A5 02 00 00 01 00 F0。

如果采 65536 字节的数据，则设置为：55 A5 02 00 00 80 00 F0。

采样速率如果是 5k，整个字节为：55 A5 00 00 00 27 0F F0。

采样通道如果是 9238 的第一通道，则设置为：55 A5 01 00 00 00 01 F0。

采样通道如果是 9238 的第二通道，则设置为：55 A5 01 00 00 00 02 F0。

这里对采样速率的设置做一个说明，这里是设置一个计数的值 27 0F，而如果为 0，采样和时钟保持一致 50M 时钟就是 50M，设置计数值后就可以改变采样频率，设置为 1 就是 25M。27 0F 换算成十进制 9999，采样速率设置是 5k，他们的关系如下。

$$\text{设置计数值} = Fclk/Fs - 1$$

Fs 是期望的采样率，Fclk 是系统时钟。

### 1.4.3 12bit 转 16bit 模块

数据采集模块 ACM9238 采集到的 12bit 数据不便于计算机存储，因为计算机对数据进行分析、存储的时候都以 8 位或 16 位数据作为统一的存储标准，所以我们需要通过数据位扩展模块（ad9238_ 12bit_to_ 16bit）将 12bit 数据转换成 16bit 数据进行存储。该模块的结构框图如下图 1-11 所示


图 1-11 数据位扩展模块结构框图

对该模块的信号说明如下表 1-6 所示。

表 1-6 数据位扩展模块信号说明表

| 信号名称 | I/O | 信号意义 |
|---|---|---|
| clk | I | 模块时钟信号 |
| ad_sample_en | I | ADC 模块数据采集使能信号 |
| ch_sel[1:0] | I | 通道设置信号 |
| ad_in1[11:0] | I | ACM9238 通道 1 的 12 位数据输入信号 |
| ad_in2[11:0] | I | ACM9238 通道2 的 12 位数据输入信号 |
| ad_out[15:0] | O | 16 位数据输出信号 |
| ad_out_valid | O | 输出数据有效信号 |

下面将编写模块实现代码。

首先，产生输出数据有效信号，将 ad_data_en 信号给到 ad_out_valid，代码 如下所示。

```
always @(posedge clk)
    ad_out_valid <= ad_sample_en;
```

然后将 ADC 采集到的无符号数据转换成有符号数据。如果采集的波形为 +5V~-5V 的正弦波，ADC 模块最终输出的数据就为 1023~0 的正弦波，但是上位机在分析数据的时候需要数据是有符号的，在这里我们进行的操作就是将 ADC 采集得到的数据加上 2048，也就是将最高位取反，最后进行分析时将最高位作为符号位。举个例子，如果 ADC 采集到的数据分别为 0 、511 、1023，将这些数据分别加上 2048 之后得到的二进制值分别为 100000000000（-0）、100111111111（-511）、 101111111111（-1023），这样将最高位作为符号位，采样的数据就变成了有符号的数据，从而可以提供给我们的上位机进行数据分析。代码如下所示。

```
Wire [11:0] s_ad_in1,s_ad_in2;
assign s_ad_in1 = ad_in1 + 2048;
assign s_ad_in2 = ad_in2 + 2048;
```

最后模块根据选择通道(ch_sel)的不同 ，输出对应通道的数据。当 ch_sel=

2'b01（0x01），输出通道 1 的数据；当 ch_sel= 2'b10（0x02），输出通道 2 的数据。ADC 采集的数据是 12 位，这里通过补 0 的方式，实现 16 位的数据输出。代码如下所示。

```
always @(posedge clk)
if(ad_sample_en && ch_sel == 2'b01)
  ad_out<={4'd0,s_ad_in1};//这样补0为了适应上位机
else if(ad_sample_en && ch_sel == 2'b10)
  ad_out<={4'd0,s_ad_in2};//
else if(ad_sample_en && ch_sel == 2'b00)
  ad_out<={4'd0,adc_test_data};
else
  ad_out <= 16'd0;
```

### 1.4.4 采样速率控制模块

采样速率控制（speed_ctrl）模块用来控制 ACM9238 的采样速率，该模块的结构框图如下图 1-12 所示。

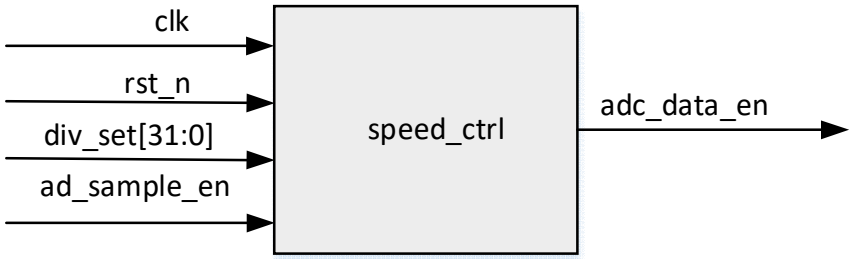


图 1-12 采样速率控制模块

对该模块的信号说明如所示。

表 1-7 采样速率控制模块信号说明表

| 信号名称 | I/O | 信号意义 |
|---|---|---|
| clk | I | 模块时钟信号 |
| reset_n | I | 模块复位信号，低电平复位 |
| ad_sample_en | I | 输入的启动采样标志信号 |
| div_set[31:0] | O | 采样频率数据控制信号， div_set=Fclk/Fs－1，Fs 是期望的采样率， Fclk 是系统时钟50M |
| adc_data_en | O | ADC 采样结果存储使能信号 |

ACM9238 模块的最大采样速率为 50M，如需使用低于时钟频率的采样速率，可以依旧给 ADC 提供 50MHz 的时钟信号，但在 FPGA 内部，对 50Msps 的采样结果数据进行抽取重采样的方法实现。比如期望以 1Msps 的采样速率采样，则只需要每间隔 50 个采样数据取一个结果存储或使用，其他 49 个数据直接舍弃，这样就能实现

1MSPS 的采样率了。下面我们将编写相应代码实现上述功能。

设置一个计数器 div_cnt，当产生采样使能信号 ad_sample_en 之后，计数器加 1，当计数值等于设置的 div_set 的时候，将计数器清零。代码如下所示：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)
    div_cnt <= 0;
else if(ad_sample_en)begin
    if(div_cnt >= div_set)
        div_cnt <= 0;
    else
        div_cnt <= div_cnt + 1'd1;
end
else
  div_cnt <= 0;
```

计数器的计数值达到 div_set 的时候，使能 ADC 采样结果存储使能信号 adc_data_en，我们将该信号输出，最终实现每隔 div_set 个采样数据取一个结果存储或使用，从而达到对 ADC 采样频率的控制。代码如下所示：

```
always@(posedge clk or negedge reset_n)
if(!reset_n)
  adc_data_en <= 0;
else if(div_cnt == div_set)
  adc_data_en <= 1;
else
  adc_data_en <= 0;
```

### 1.4.5 DDR2 控制器（DDR2_control）

该控制器模块和读写四个 FIFO（两个写 FIFO，两个读 FIFO）共同组成了 DDR2_control 模块其中写 FIFO 负责将采集的数据缓存，由 DDR2 控制器读取并控制其存储到片外存储设备 DDR2 中，读 FIFO 负责对 DDR2 中要读出的数据缓存并在 DDR2 控制器的控制下输出。为 DDR2 控制器加入四个 FIFO，该设计很好地解决了在某些特殊的时刻，有些读或写会被忽略掉，而且数据的写或读不能连续对数据流进行缓存，只能间歇式的读或写 DDR2 数据，导致数据存储或读取遗漏的问题，这里就不详细的进行讲解。

### 1.4.6 状态控制模块（state_ctrl）

状态控制模块是本系统的核心控制模块，其依据 uart_byte_rx 模块给出的数据采集量信息 set_sample_num 、start_sample 、DDR2 控制器模块 wrfifo_full 信号对 DDR2 控制器模块的 wdfifo 的数据写入进行协调控制，同时对从 DDR2 控制器模块读数据进行控制，对从 DDR2 控制器模块读出的数据进行转换并控制输出到 uart_byte_tx 模块，状态控制模块如图 1-13。

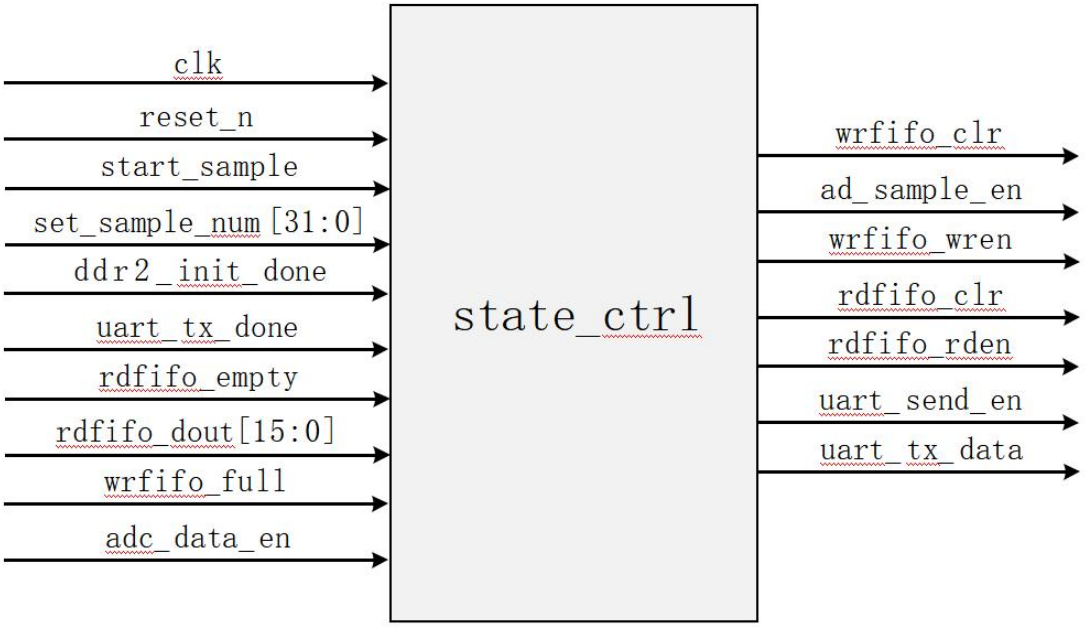


图 1-13 状态控制模块

该模块的接口功能说明如下表 1-8 所示。

表 1-8 状态控制模块接口功能描述

| 端口名 | 端口类型 | 描述 |
|---|---|---|
| clk | input | 系统时钟 50MHz |
| reset_n | input | 模块复位信号 |
| start_sample | input | ACM9238 模块开始采样标志信号 |
| set_sample_num | input [15:0] | 设置的采样深度，16 位计数，最大65535 |
| ddr2_init_done | input | ddr 初始化完成信号 |
| uart_tx_done | input | 串口发送完成标志信号 |
| rdfifo_empty | input | 读FIFO 的读空标识信号，用于标识当前FIFO 是否为空（即FIFO 内有无数据） |
| rdfifo_dout | input [15:0] | 读FIFO 的读数据输出，数据位宽为 16 位 |
| wdfifo_full | input | 写 FIFO 的写满标识信号，用于标识当前FIFO 是否有被写满 |
| adc_data_en | input | ADC 采样结果存储使能信号 |
| wrfifo_clr | output | wrfifo 清零信号 |

| | | |
|---|---|---|
| ad_sample_en | output | adc 采集的使能信号 |

在这个模块中，定义了 11 个状态， 如下所示：

```
localparam IDLE  = 4'd0;        //空闲状态
localparam DDR_WR_FIFO_CLEAR = 4'd1; //写fifo清0状态
localparam ADC_SAMPLE = 4'd2;   //ADC采样状态
Localparam DDR_RD_FIFO_CLEAR = 4'd3; //读fifo清0状态
Localparam DATA_SEND_DELAY1 = 4'd4; //延时过渡状态1
Localparam DATA_SEND_DELAY2 = 4'd5; //延时过渡状态2
Localparam DATA_SEND_LOW_START = 4'd6; //低8位数据开始发送状态
Localparam DATA_SEND_LOW_WORKING = 4'd7; //低8位数据发送进行状态
Localparam DATA_SEND_HIGH_START = 4'd8; //高8位数据开始发送状态
Localparam DATA_SEND_HIGH_WORKING = 4'd9; //高8位数据发送进行状态
Localparam DATA_SWITCH = 4'd10; //判断发送是否完成状态
```

第一步：程序上电后，状态机进入空闲状态。当 DDR2 初始化完成并且收到开始采样的指令，则进入状态 1。

第二步：进入状态 1 后开始清除写 fifo 内的原始数据。进入清写 fifo 状态后，FPGA 发送三拍的拉高信号清写 fifo 指令，并且给出 10 拍的基本延时保证，当写 fifo 内的原始数据清除完毕后，fifo 内部会将写端 fifo_full 的信号拉低。如果收到写端 fifo_full 信号拉低，说明 fifo 已经不满， FPGA 可以开始向 fifo 内传递从 ACM9238 采集到的数据。

```
DDR_WR_FIFO_CLEAR: //1
        begin
        if(!wrfifo_full && (wrfifo_clr_cnt==9))
            state<=ADC_SAMPLE;
        else
            state<=DDR_WR_FIFO_CLEAR;
        end
```

清除写 fifo 的指令，由三拍延时信号拉高提供。

```
/*初始化成功后， 进行一次清fifo,如果进入了SDRAM_WR_FIFO_CLEAR状态，则在 wrfifo_clr_cnt为0,1或2时， 清写fifo置1，否则wrfifo_clr为0*/
always@(posedge clk or posedge reset)begin
if (reset)
    wrfifo_clr<=0;
else if(ddr2_init_done==1'b0)
    wrfifo_clr<=1'b1;
else if(state==DDR_WR_FIFO_CLEAR)
      begin
```

```
                if(wrfifo_clr_cnt==0||wrfifo_clr_cnt==1||wrfifo_c
            lr_cnt==2) wrfifo_clr<=1'b1;
          else
                wrfifo_clr<=1'b0;
        end
   else
     wrfifo_clr<=1'b0;
  end
```

在程序中 reg[4:0]wrfifo_clr_cnt 信号为写 fifo 清零的状态计数和保持，当进入写 fifo 清零状态后，首先开始计数，先保证计数完成，再等待 wrfifo_full（写端 fifo 满信号）的信号拉低，拉低后，表示可以往 fifo 里写入数据，此时进入下一个状态。在清空（复位）fifo 的时候，fifo 的 full 信号会变高，可以认为在复位 fifo 时是不允许对 fifo 进行写操作的，即使写也是不可靠的，等 fifo 的复位结束后，full 信号会变低，就允许对 fifo 进行写操作。清写端 fifo 的控制信号是由计数器(在前 3 个计数值将清除控制信号拉高)产生 3 个时钟周期的高电平脉冲。

第三步：设定采样参数。采样数据的速率，和 FPGA 的非 DDR2 工作时钟频率保持一致，为每秒传递 50M 的 16 位数据，采样数据的个数设定，项目开发时准备了两种方案，一种是通过 ISSP 进行设定，这种方案较初级，我们工程中采用的是更高级一点的，就是前面提到过的，通过串口指令的方式进行设定。

第四步：进入数据采样状态即状态 3。状态 wrfifo_full 拉低，在这个状态，当产生 adc_data_en 信号时，计数器开始计数，当发到和设定的采样需求个数相同时，跳转进入下一个状态，开始清除读 fifo。

```
always @ (posedge clk or negedge reset)
if (!reset)
    adc_sample_cnt<=32'd0;
else if(state==ADC_SAMPLE)begin
    if(adc_data_en)
        adc_sample_cnt<=adc_sample_cnt+1'b1;
    else
        adc_sample_cnt<=adc_sample_cnt;
    end
else
    adc_sample_cnt<=32'd0;
```

状态机跳转：

```
ADC_SAMPLE :   begin
```

```
if (adc_sample_cnt>=set_sample_num - 1'b1)
        state <= DDR_RD_FIFO_CLEAR;
else
        state <= state;
end
```

第五步：一旦进入读 fifo 清零状态，我们做和前面写 fifo 清零相同的操作，发出三拍的清零指令，同时保证一个 10 拍的基本延时。等延时结束并且接收到读 fifo 反馈的 rdfifo_empty 信号后，进入下一个状态。

```
    DDR_RD_FIFO_CLEAR: //1
            begin
            if(!rdfifo_empty && (rdfifo_clr_cnt == 9))
                state <= DATA_SEND_DELAY1;
            else
                state <= DDR_RD_FIFO_CLEAR;
            end
```

清除读 fifo 的指令，由三拍延时信号拉高提供。

```
    always@(posedge clk or posedge reset)begin
    if (reset)
        rdfifo_clr<=0;
    else if(ddr2_init_done==1'b0)
        rdfifo_clr<=1'b1;
    else if(state==DDR_RD_FIFO_CLEAR)
          begin
                  if(rdfifo_clr_cnt==0||rdfifo_clr_cnt==1||rdfifo_c
              lr_cnt==2) wrfifo_clr<=1'b1;
            else
                  rdfifo_clr<=1'b0;
          end
    else
      rdfifo_clr<=1'b0;
   end
```

第六步：下一个状态是延时过渡状态 1，在这一拍，我们可以开启向读 fifo 发送一个脉冲的数据读使能信号(rdfifo_rden==1)。再进入延时过渡状态 2。

```
   DATA_SEND_DELAY1   :   begin
           state <= DATA_SEND_DELAY2;
           rdfifo_rden <= 1'b1;
   end
```

第七步：进入延时过渡状态 2 后，关闭读 fifo 的数据读使能信号(rdfifo_rden==0)。进入低 8 位数据开始发送状态，则通知串口可以向外发送低 8 位数据。接下来，按照串口的节奏，进行数据的发送。

```
DATA_SEND_DELAY2   :   begin
        state <= DATA_SEND_LOW_START;
        rdfifo_rden <= 1'b0;
end
```

关于串口的数据发送，在设计之初也有两种预备方案，一种是通过总节拍数控制串口发送的结束，另一种，是利用串口发送完成后给出的 tx_done 信号，和提取数据的判定条件形成关联，每当一个串口字节发送完成后，利用串口给出的 tx_done 信号，切换到下一个字节的发送开始状态。这个 tx_done 信号直接控制和协调下一个小循环读 fifo 的 16 位数据提取工作，提取的数据缓存工作，缓存的数据拆解字节到串口寄存器的工作。在本工程中，我们从节约串口时间 成本的角度考虑，使用了方案 2，而没有纯粹追求更简单的串口控制方法。同时，单个字节发送完成的信号，也可以作为从发送低 8 位到发送高 8 位的判定信号条件。

接下来的几个步骤描述的是串口数据位的发送。其对应的逻辑关系图如下图 1-14 。由于 rdfifo_rden 的脉冲信号要到下一拍才能让数据从 rdfifo 中读出，而读出的数据发送到串口的 send_en 脉冲信号要等数据稳定后下一拍生效才能发出正确的数据，所以，每个 16 位的数据，完成从 fifo 中提取到发送到串口，至少要三拍延时。

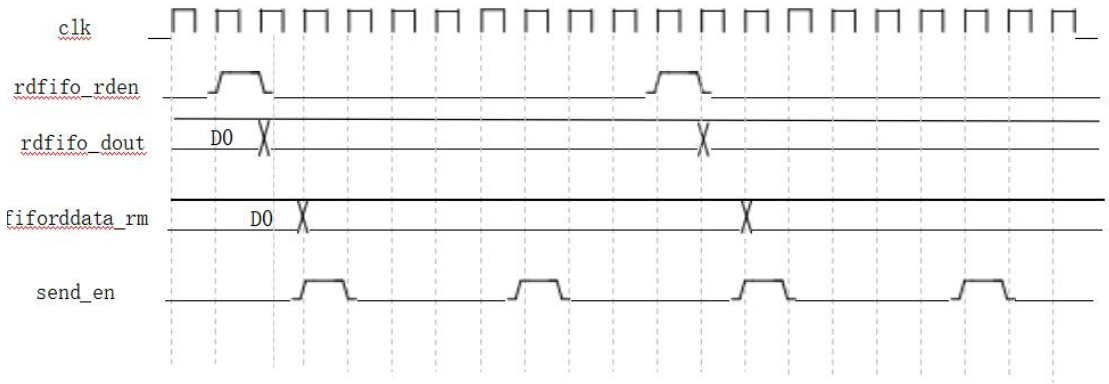


图 1-14 串口数据发送逻辑关系图

上图为：在状态机使用条件下，从读 fifo 中读取数据，数据发送到串口的三拍延时的触发启动流程。到最后，就是状态 10，状态 10 是个用来判断是否完成数据块发送的状态，如果确实数据块发送完成，则回到 IDLE 状态大循环收口关闭，如果没有达到数据总共发送次数，则表明没有发送完，回到 delay1 状态，发送下一个数据，本轮 16 位串口发送小循环收口关闭，进入下一个小循环。

## 1.5 板级验证

### 1.5.1 系统所需硬件

1、AC6103 开发板
2、ACM9238 模块
3、电源线
4、下载器
5、串口线
6、信号发生器

### 1.5.2 硬件连接

1、连接电源线
2、连接好下载器
3、连接好串口
4、将 ACM9238 模块连接到 GPIO1 上。
完成上述步骤即可以打开电源开关。

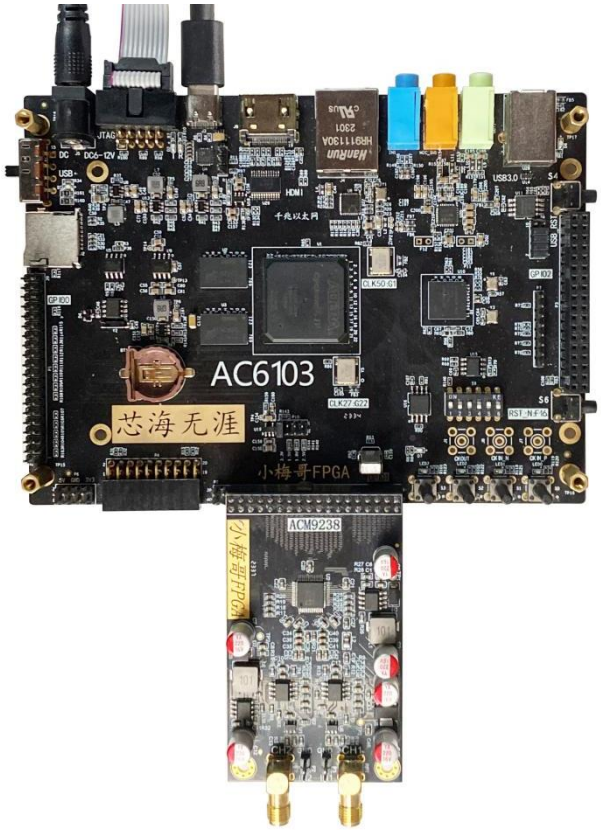


图 1-15 硬件连接图

### 1.5.3 串口调试助手采集数据并分析

开 sscom5.13.1 串口调试工具，依次设定好 com 端口，波特率，然后打开串口。点击多字符串，点击勾选三个指令栏，填入指令。

采 65536 字节的数据，则设置为： 55 A5 02 00 00 80 00 F0

采样通道为第一通道，则设置为： 55 A5 01 00 00 00 01 F0

采样的速率为 50M，指令设置为： 55 A5 03 00 00 00 00 F0

启动采样，指令设置为： 55 A5 00 00 00 00 00 F0

清空计数器和接收区后，从上到下依次点击勾选的 1 ，2 ，3 条数据串发送按钮，数据设定完成后开始采集，串口调试助手设置界面如图 1-16。

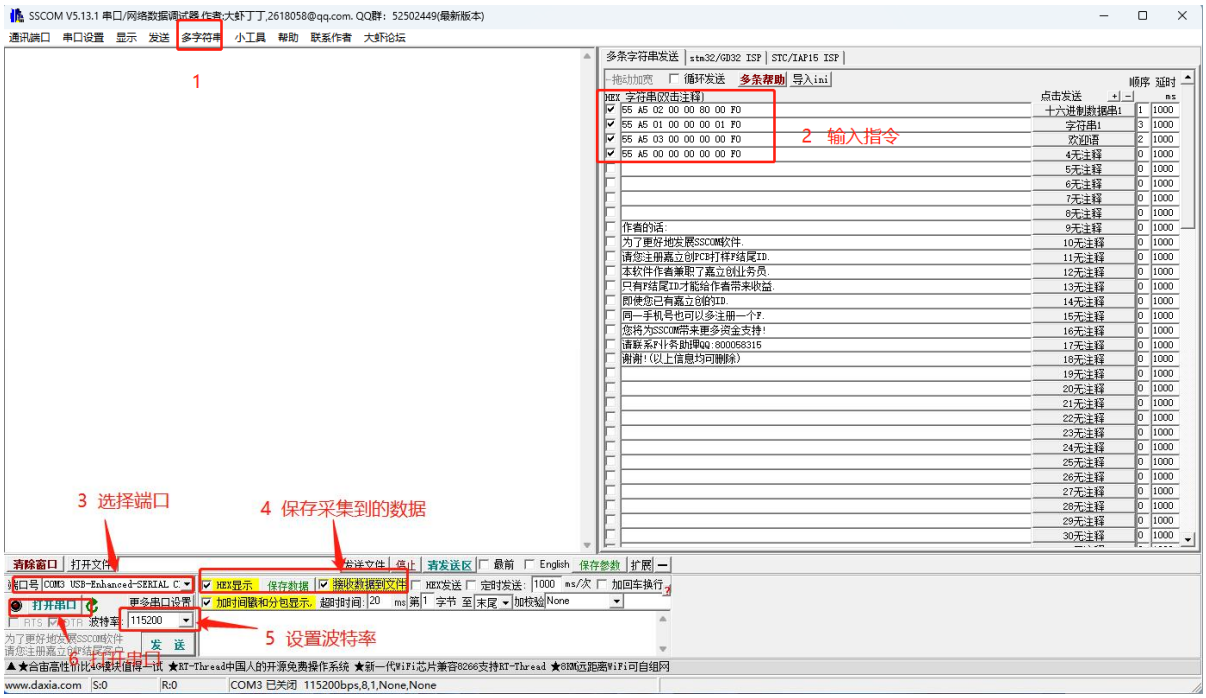


图 1-16 串口调试设置

通过调试窗口可以看到，采集的数据量和下发的指令是匹配的都为 65536 ，对采集的数据做进一步的分析可以验证数据采集是否符合期望，可对采集到的数据利用 MATLAB 进行绘图分析。

通过串口调试助手可以判断采集的数据量是否准确，采样得到的数据是否是和信号发生器输出一致，仅凭人工，无法完成这个分析工作。因此我们需要借助 matlab 的绘制函数图形的功能。

打开 ADCdata_to_wave_v2_2.m ，MATLAB 采样结果数据处理函数如图 1-17：

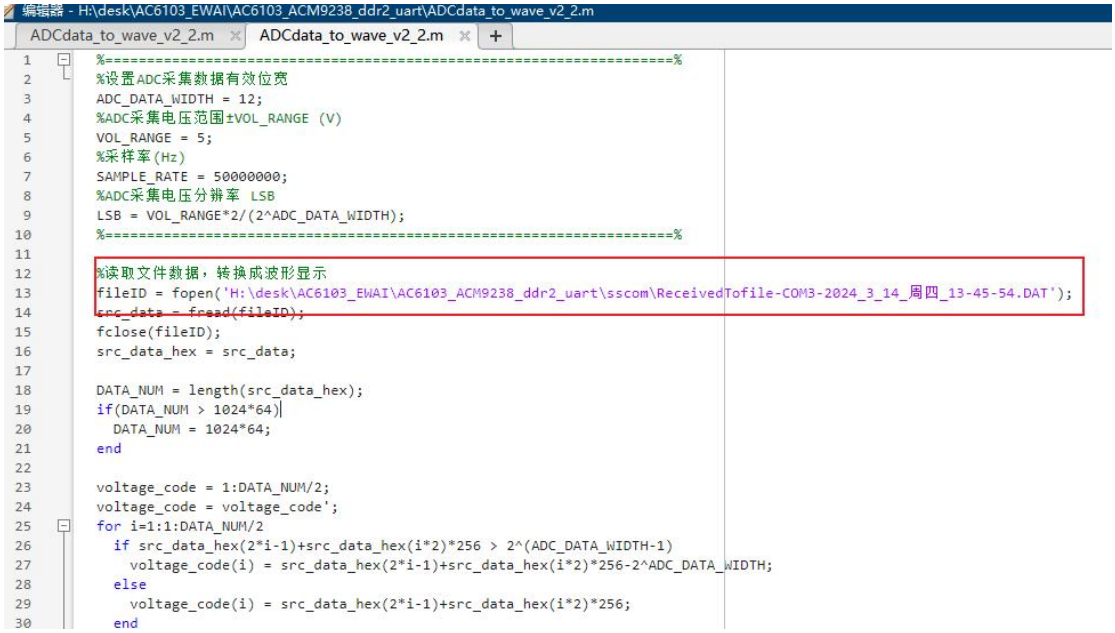


图 1-17 MATLAB 数据分析程序

点击串口助手保存数据会产生两个数据文件，第一个是 DAT 格式的数据文件，第二个是文本格式的数据文件，读者可根据自身需求对不同格式的文件进行调用，本次实验调用的是 DAT 格式的数据文件。采集数据文件路径都与安装的串口调试助手的路径有关，在串口调试窗口中点击数据保存时会有显示，下图显示的是 sscom5.13.1 串口调试工具安装于桌面的数据保存路径。


图 1-18 保存数据位置信息

文件为二进制格式 DAT 文件，调用时可复制文件到 MATLAB 安装路径 下，这样修改文件名就可以进行图像绘制。 DAT 文件如图 1-19 所示。

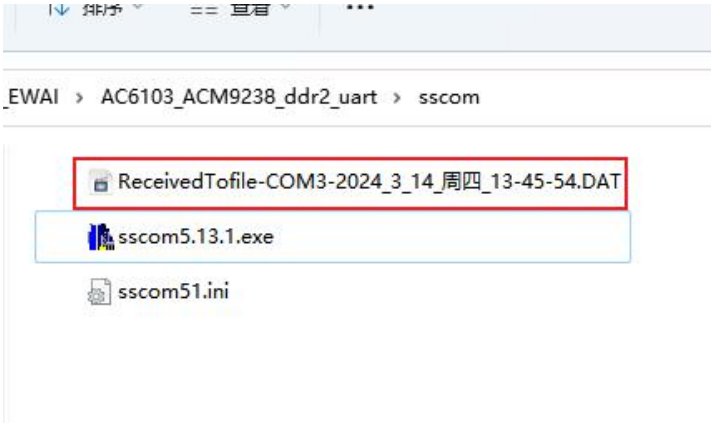


图 1-19 DAT 文件

开matlab 代码的工程文件，修改好文件路径，进行保存后点击运行就可 以进行图形的绘制。在进行新一次的数据绘制分析前要关闭前一次的图形绘制窗口，否则会导致新数据的图形绘制无法加载。输出得到的波形如图 1-20。

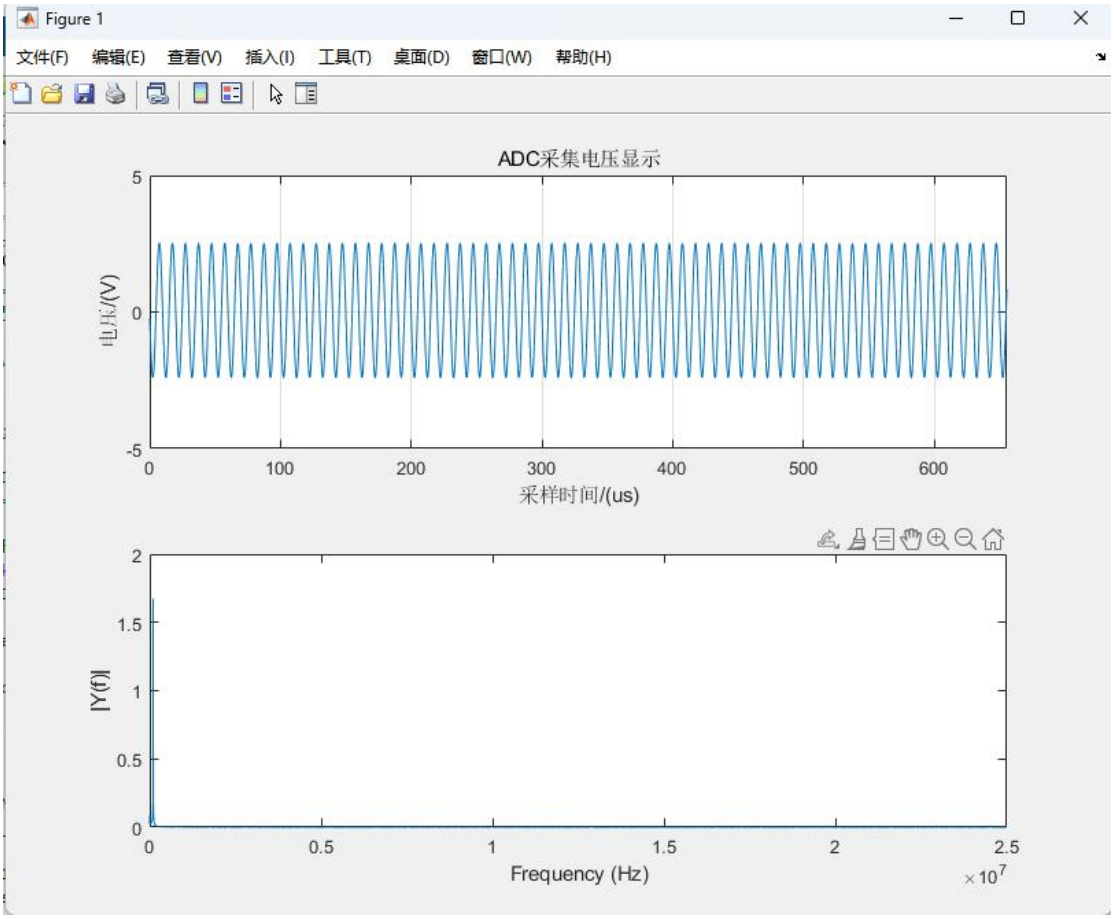


图 1-20 绘制的数据波形图

这样，通过观察绘制的数据图形就可以对上板实验采集数据进行验证。通过分析，绘制数据与信号发生器和在示波器上显示的数据一致，数据采集无误，采集系统功能验证符合设计要求。

## 1.5.4 数据采集上位机采集数据

打开我们提供的数据采集上位机“小梅哥控制台 For ADC 采集”，初始界面如下图 1-21 所示。

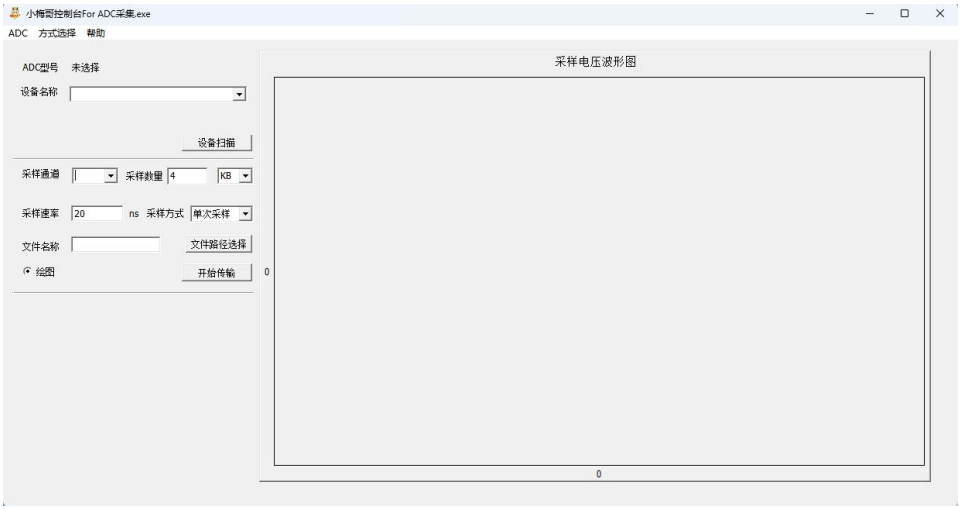


图 1-21 数据采集上位机初始界面

然后依次选择 ADC->ACM9226，方式选择串口，采样方式选择循环采样，这样上位机就能源源不断的采集数据。如下图 1-22 所示。

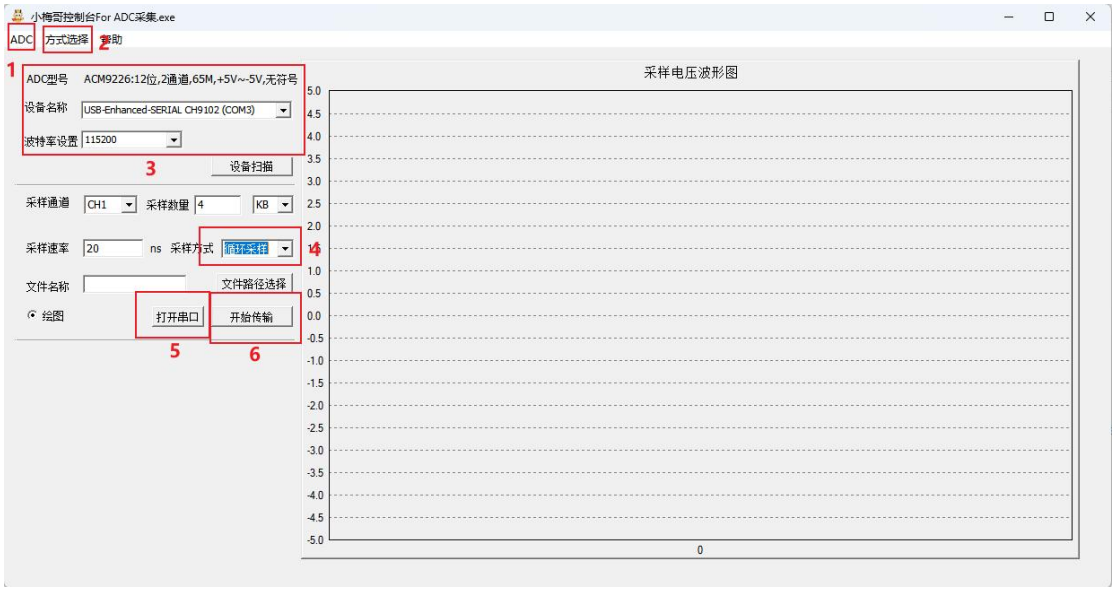


图 1-22 选择方式和ADC 型号

最后按照默认，打开串口，点击开始传输，如下图 1-23 所示，需要注意的是串口助手和数据采集上位机不能同时打开串口，否则会出现串口打开失败。

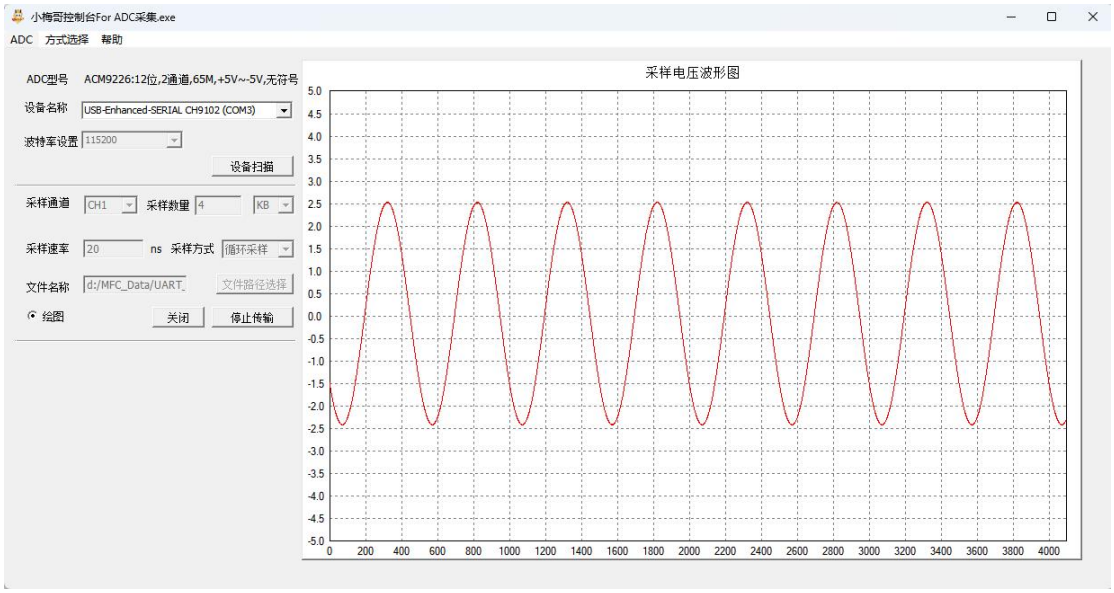


图 1-23 上位机显示数据采集波形图

从上图可以看出，采集的波形为正弦波，且没有杂波， 需要注意的是波形图的横坐标对应的不是频率，而是采样数量，数据默认保存在 d:/ MFC_Data 文件夹下，用户也可以通过 MATLAB 对采集的数据进行进一步分析。

## 1.6 总结

本实验实现了通过串口下发指令对 ACM9238 模块采集数据的数据量、采集通道、采样速率进行合理设置，利用 DDR2 对采集数据进行缓存，再通过串口对采集到的数据发送到 PC 端的数据采集系统。介绍了 ACM9238 模块的功能特点、使用方式和硬件连接方法，工程设计的思路，串口发码程序向 FPGA 发布工作控制指令的设计和应用方法，利用 matlab 进行数据绘图的操作方法。