



AC820 开发板

基于 TCP 传输系统的 ACFL3432 设置和数据采集与显示说明

寄存器配置 / IDELAY 自动训练 / DMA 缓存 / TCP 传输

武汉芯路恒科技有限公司

教学产品项目组 编写

修订说明:

V1.0.0	2026-09-23	第一版本发行

版权所有 © 2025 武汉芯路恒科技有限公司



、小梅哥均为武汉芯路恒科技有限公司注册商标，本手册中提到的其他任何商标，其所有权利属其拥有者所有。未经本公司书面许可，任何单位和个人都不得擅自摘抄、复制、翻译本文档内容的部分或全部，不得以任何形式传播。

免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除芯路恒在其产品的销售条款和条件中声明的责任之外，武汉芯路恒概不承担任何法律或非法律责任。武汉芯路恒对芯路恒产品的销售和 / 或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。芯路恒对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，芯路恒保留修改文档中任何内容的权利，恕不另行通知。芯路恒不承诺对这些文档进行适时的更新。

联系我们

厂家名称：武汉芯路恒科技有限公司

联系地址：武汉东湖新技术开发区民族大道 331 号福成科技园 1 号楼第六层第 6018 号

联系电话：027-59265620

技术网站：fpga.cn

微信公众号（提供在线客服）：小梅哥电子

在线商城（淘宝）：<https://xiaomeige.taobao.com>

在线商城（天猫）：<https://xiaomeigesm.tmall.com/>

目 录

第 1 章 基于 TCP 传输系统的 ACFL3432 设置和数据采集与显示说明.....	2
1.1 ACFL3432 模块简介.....	2
1.1.1 FMC 接口与双通道数据组织	2
1.2 整体系统设计.....	4
1.2.1 采样数据通路	5
1.2.2 控制通路	5
1.3 代码设计.....	6
1.3.1 PL 端逻辑系统设计	6
1.3.2 PS 端应用程序设计.....	29
1.4 板级验证.....	38
1.4.1 实验所需硬件	38
1.4.2 修改电脑 IP 地址	38
1.4.3 代码烧录	41
1.4.4 功能验证	42
1.5 思考与总结.....	50

第1章 基于 TCP 传输系统的 ACFL3432 设置和数据采集与显示说明

工程源码	-- AC820 开发板 -- AC820_ACFL3432_TCP_ytw8531
相关视频课程	
参考官方文档	
特别说明	本工程每轮通过 DMA/TCP 传输一路数据，可选择通道 1、通道 2 或 FPGA 测试计数。接收逻辑同时恢复 A、B 两路数据，但不提供同一轮双通道打包传输。

章节导读

本章使用 AC820 开发板搭载 ACFL3432 采集模块，完成 CM3432 寄存器配置、LVDS 双沿接收、IDELAY 自动扫 tap、采样数据缓存和以太网 TCP 传输。PC 端通过控制命令设置采集参数，接收 DDR 中的采样数据，并按正确的数据格式绘制波形。

1.1 ACFL3432 模块简介

ACFL3432 模块采用 CM3432 双通道 14 位高速 ADC，最高采样率为 250 MSPS。

两路模拟输入分别经过前端信号调理后送入 ADC，模块按 $\pm 5\text{ V}$ 外部输入范围使用。ADC 芯片内部量程与模块接口量程属于两个不同位置的参数，电压换算时需要考虑模拟前端增益。ACFL9253 模块图如下图所示。

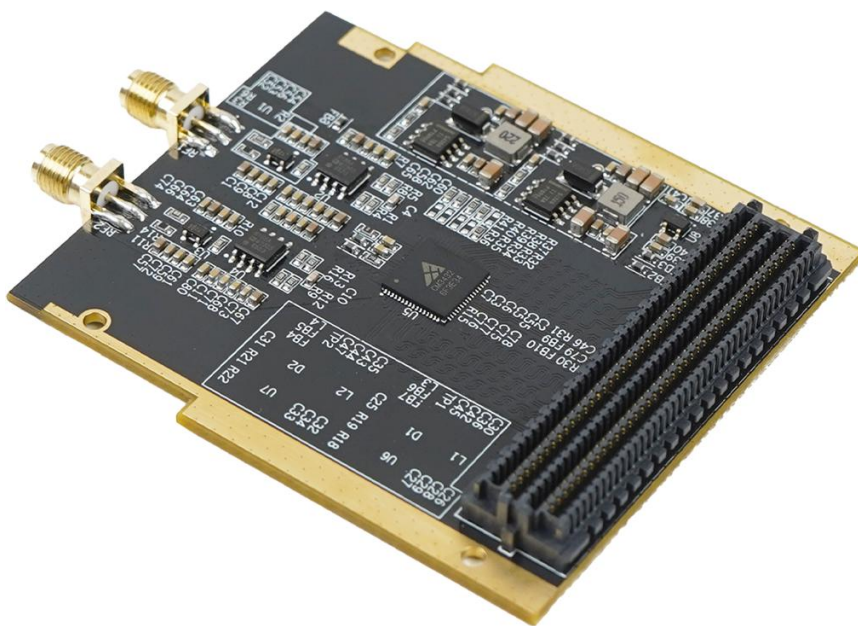


图 1-1 ACFL3432 模块实物图

1.1.1 FMC 接口与双通道数据组织

ACFL3432 通过 FMC_LPC 接口与开发板连接。主要数字信号包括 FPGA 送往 ADC 的差分采样时钟、ADC 返回的数据时钟 DCO、14 对并行 LVDS 数据，以及 SPI 与 PDWN 控制信号。使用模块前应先确认接口方向和 2.5 V 供电选择。

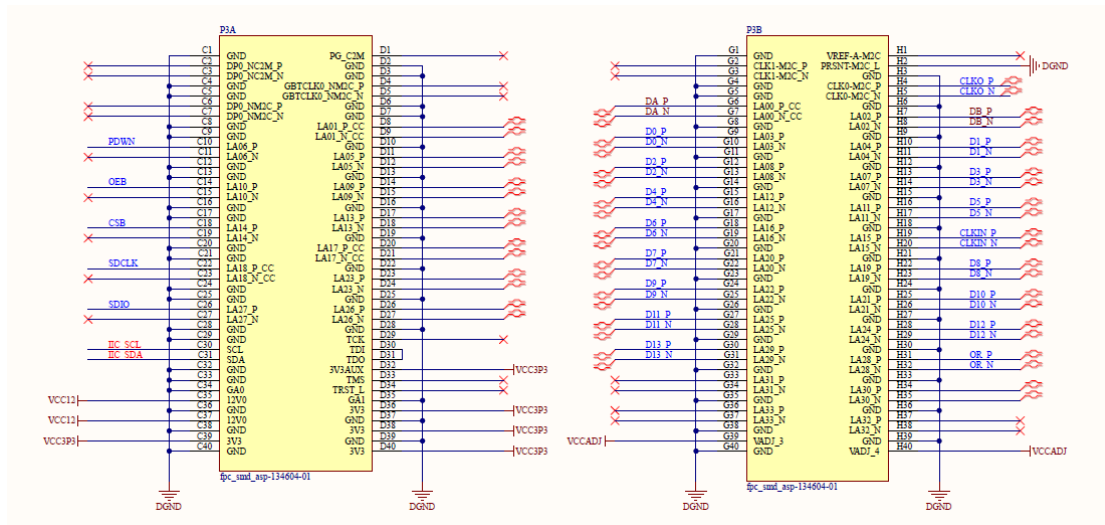


图 1-2 ACFL3432 模块 FMC_LPC 接口

CM3432 将两个通道的数据交织在同一组 14 对数据线上。FPGA 在返回时钟的两个边沿分别接收数据：IDDR 的 Q1 汇成 A 路 14 位总线，Q2 汇成 B 路 14 位总线。采用 SAME_EDGE_PIPELINED 后，两路数据对齐到同一个上升沿输出，再经过一级公共寄存器送往训练和采集逻辑。

表 1-1 主要信号说明表

信号或接口	方向	作用
adc_clk_p / adc_clk_n	FPGA → ADC	提供默认 250 MHz 差分采样时钟
adc1_clk_p / adc1_clk_n	ADC → FPGA	返回 DCO，作为源同步接收时钟
adc1_data_p/n[13:0]	ADC → FPGA	14 对并行 LVDS，双沿承载 A/B 数据
SPI_CS / SPI_SCLK / SPI_IO	双向配置接口	执行寄存器写入与读回
PDWN	FPGA → ADC	控制上电保持与唤醒过程

当 CM3432 芯片模拟输入端接-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，根据对 0x14 寄存器设置不同，CM3432 芯片转换输出格式也有不同，比如当前工程中对 0x14 寄存器设置如下：

```
.....
5'd6: lut_data <= {16'h0014, 8'h05};
...
```

根据手册中对 0x14 寄存器参数解释，CM3432 支持三种输出编码：二进制移码；二进制补码；格雷码。

OUTPUT MODE (LOC, 0x14)

位	名称	访问类型	复位	描述
7:5	RSV	RW	000	保留位。
4	OUTPUT ENABLE	RW	0	通道 ADC 输出使能。若只关闭一个通道，则关闭通道的输出会重复另一通道。
3	RSV	RW	0	保留位。
2	OUTPUT INVERT	RW	1	输出反向。 0: 反向输出 1: 正常输出
1:0	OUTPUT FORMAT	RW	01	输出格式。 00: 二进制移码 01: 二进制补码 10: 格雷码 11: 保留

图 1-3CM3432 芯片手册中对于 0x14 寄存器解释

当前工程中对 0x14 寄存器设置为 0x05，也就是当前工程中 CM3432 被配置成了 14 位二进制补码，因此采集数据应按“有符号 14 位数据”使用，转换波形如下图所示：

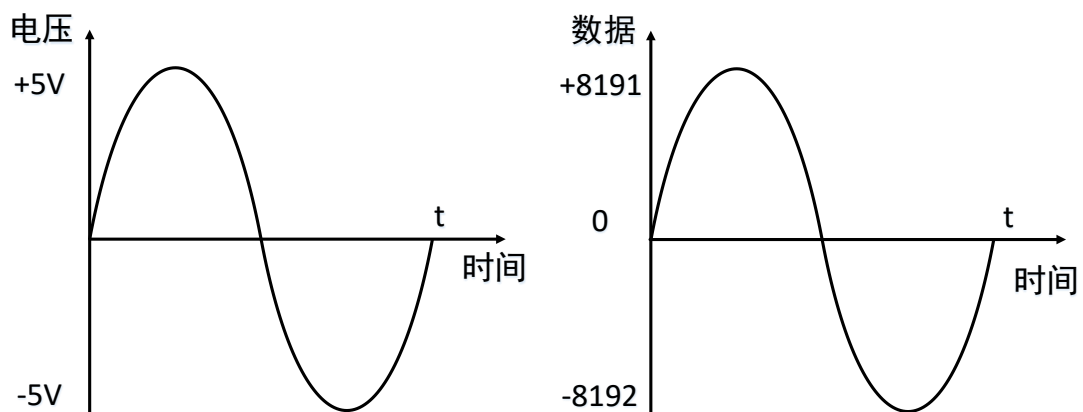


图 1-4CM3432 正弦波模拟电压值（左）、数据（右）

1.2 整体系统设计

系统由高速采集前端和数据传输后端组成。前端先建立稳定的接收时序，再输出有效采样数据；后端按设定长度将数据写入 DDR，全部采集完成后通过 TCP 发送。采样速度与网络发送速度不要求在同一时刻相等。

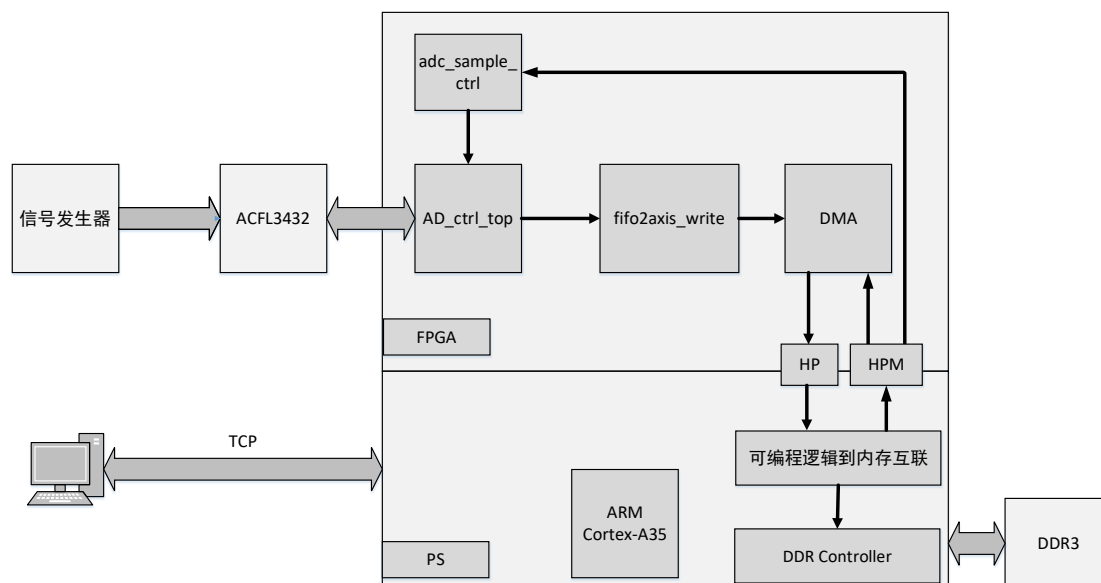


图 1-5ACFL3432 采集、QT 上位机显示示意图

1.2.1 采样数据通路

表 1-2 采样数据通路表

数据经过的环节	本环节完成的工作
ACFL3432 → LVDS 接收	CM3432 输出交织数据，经 IBUFDS、IDELAYE2、IDDR 恢复 A/B 两路 14 位样点
通道选择 → 16 位数据	选择一路数据，在高两位补 0，形成 16 位传输容器
异步 FIFO → AXI4-Stream	16 位写入、128 位读出，完成跨时钟域缓存及每 8 个样点的打包
AXI DMA → PS DDR	使用 S2MM 通道，按段写入指定 DDR 缓冲区
PS → TCP → PC	lwIP 分片发送原始字节，PC 累计接收、解码和绘图

1.2.2 控制通路

上位机发送 8 字节命令，PS 程序负责组帧、解析和任务调度。采样参数通过自定义 AXI 控制 IP 送入 PL；ADC 寄存器访问由 PL 转换为 SPI 事务，并通过状态寄存器返回处理结果。

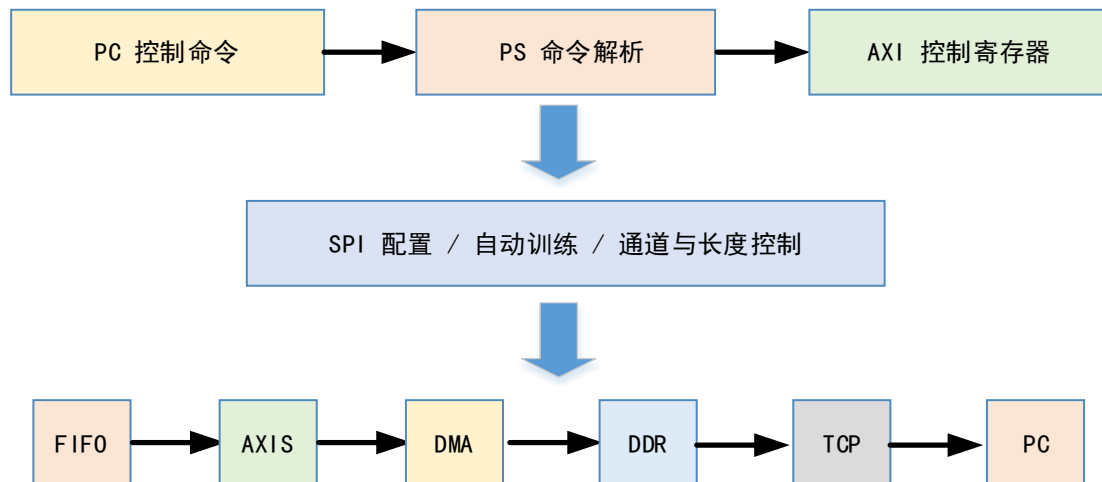


图 1-6控制通路

开发板上电后先完成 PDWN 时序、SPI 初始化和自动扫 tap，随后退出棋盘格模式。用户连接 TCP 服务，读取训练结果，再设置通道与采集字节数并启动采样。修改会触发重训练的寄存器后，应等配置事务完成，再开始下一轮采集。

1.3 代码设计

1.3.1 PL 端逻辑系统设计

1.3.1.1AD 控制顶层模块

Vivado 工程顶层为 top，定义在 top_tcp.v 中。顶层连接 AD_ctrl_top_tcp 与 design_1_wrapper：前者管理 ADC 配置、训练和采样输出，后者包含 Zynq PS、时钟、AXI 控制 IP、FIFO 数据流转换和 DMA。

表 1-3PL 主要模块

模块/文件	主要功能
AD_ctrl_top_tcp.v	连接上电、命令仲裁、自动训练、状态反馈和采样门控
ACFL3432_ctrl.v	ADC 时钟转发、DCO 接收、14 位 DDR 接收及训练触发
cm3432_powerup.v	PDWN 保持、唤醒等待和 SPI 初始化允许
lut_config.v、spi_ctrl.v	上电配置表、运行时寄存器读写及影子寄存器提交
cm3432_ddr_data_bit.v	单个数据位的差分输入、可编程延时与双沿采样
cm3432_tap_trainer.v	检查两路棋盘格图案和相邻样点交替关系
idelay_tap_window_trainer.v	扫描 tap 0~31，记录最长连续窗口并装载中心值
cm3432_runtime_retrain.v	管理 0x000B 配置变化后的自动重训练

模块/文件	主要功能
ad_14bit_to_16bit.v	测试源/模拟通道选择和 16 位数据输出
fifo2axis_write_ADC.v	异步 FIFO、128 位数据打包、握手和 TLAST

1.3.1.2 时钟与差分信号处理

本 TCP 工程由 PS 输出 100 MHz FCLK_CLK0，Clocking Wizard 在此基础上产生 ADC 所需的三个时钟。ADC 返回的 DCO 单独形成源同步接收时钟域，FIFO 用于连接 DCO 域与 AXIS 域。

表 1-4 时钟信号

时钟	频率	使用位置
FCLK_CLK0	100 MHz	AXI 互连、自定义控制 IP、DMA、FIFO 读侧
clk_out1 / clk_50M	50 MHz	PDWN、SPI、命令处理及重训练控制
clk_out2 / clk_250M	250 MHz	转发到 ACFL3432 的采样时钟
clk_out3 / idelay_ref_clk	200 MHz	IDELAYCTRL 校准参考
adc1_clk	由 ADC 返回	IDDR 接收、数据寄存、扫描 tap、FIFO 写侧

7

1.3.1.2.1 差分时钟输出

ACFL3432_ctrl 先使用 ODDR 产生与输入时钟对应的输出波形，再通过 OBUFDS 驱动 ADC 差分时钟引脚。ODDR 的上升沿输出 1、下降沿输出 0。

```

ODDR #(
    .DDR_CLK_EDGE ("SAME_EDGE"),
    .INIT          (1'b0),
    .SRTYPE        ("ASYN")
) ODDR_adc_clk_forward (
    .Q (adc_clk_forwarded),
    .C (clk_100),
    .CE (1'b1),
    .D1 (1'b1),
    .D2 (1'b0),
    .R (rst),
    .S (1'b0)
);

OBUFDS #(
    .IOSTANDARD ("LVDS_25"),
    .SLEW        ("SLOW")
) OBUFDS_adc_clk (

```

```
.O (adc_clk_p),  
.OB (adc_clk_n),  
.I (adc_clk_forwarded)  
);
```

这里 ACFL3432_ctrl 的 clk_100 端口接入的是 clk_250M，即 250 MHz。判断时钟频率应结合顶层端口连接与 IP 配置。外部低有效复位 reset_n_0 接入 AD 控制顶层，BD 的 AXI 复位链路另由复位模块管理。

1.3.1.2.2 返回 DCO 与单 bit LVDS 接收链路

CM3432 返回的数据时钟经过 IBUFDS 转换为单端信号，再由 BUFG 送入 FPGA 全局时钟网络。14 个数据位和后级数据寄存器使用同一个 adc1_clk，形成统一的接收时钟域。

```
IBUFDS #(  
  .DIFF_TERM ("TRUE"),  
  .IBUF_LOW_PWR ("FALSE"),  
  .IOSTANDARD ("LVDS_25")  
) IBUFDS_adc1_clk (  
  .O (adc1_clk_ibuf),  
  .I (adc1_clk_p),  
  .IB (adc1_clk_n)  
)  
);  
  
BUFG BUFG_adc1_clk (  
  .I (adc1_clk_ibuf),  
  .O (adc1_clk)  
)  
);
```

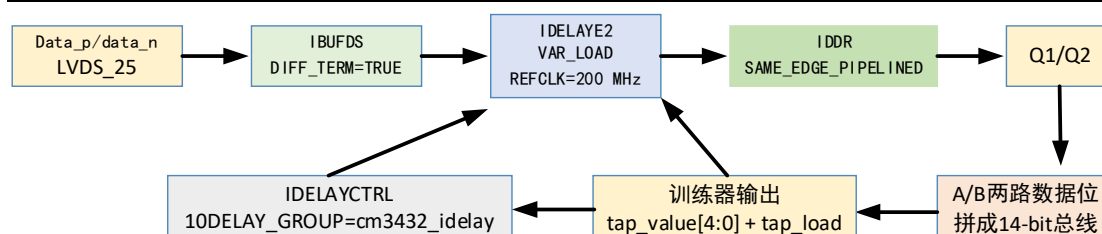


图 1-7 单 bit LVDS 接收与公共 tap 控制链路

数据输入不是只做一次差分转单端。每个数据位先由 IBUFDS 接收，再通过 IDELAYE2 调整到达 IDDR 的时间位置，最后用 IDDR 的两个边沿取得 A、B 通道数据。

tap 是输入延时单元的离散设置值。本工程以 5 位数据表示 tap，范围为 0~31。14 个数据位共用 train_tap_value 和 train_tap_load，因此自动扫描寻找的是整条并行数据总线的公共可用窗口。

注意：IDELAYCTRL 的 RDY 表示延时单元校准就绪；training_pass 表示 ADC 测试图案检查通过。两者由不同环节产生，应分别观察。

1.3.1.2.3 IDELAYE2 装载与 IDDR 双沿采样

cm3432_ddr_data_bit.v 对每一个数据位实例化相同的接收结构。IDELAYE2 采用 VAR_LOAD 模式，tap_load 有效时，将 tap_value 装入延时单元。REFCLK_FREQUENCY 设置为 200.0，与 IDELAYCTRL 的参考时钟相对应。

```
IDELAYE2 #(
    .CINVCTRL_SEL      ("FALSE"),
    .DELAY_SRC          ("IDATAIN"),
    .HIGH_PERFORMANCE_MODE ("FALSE"),
    .IDELAY_TYPE        ("VAR_LOAD"),
    .IDELAY_VALUE       (IDELAY_VALUE),
    .PIPE_SEL           ("FALSE"),
    .REFCLK_FREQUENCY   (200.0),
    .SIGNAL_PATTERN     ("DATA")
) IDELAYE2_adc_data (
    .DATAOUT      (data_delay),
    .IDATAIN      (data_ibuf),
    .DATAIN       (1'b0),
    .C             (capture_clk),
    .CE           (1'b0),
    .CINVCTRL     (1'b0),
    .CNTVALUEIN   (tap_value),
    .CNTVALUEOUT  (),
    .INC          (1'b0),
    .LD           (tap_load),
    .LDPIPEEN     (1'b0),
    .REGRST       (rst)
);
```

IDDR 使用 SAME_EDGE_PIPELINED，将两个边沿接收的结果对齐到同一上升沿，便于后续使用完整的一组 A/B 数据。相关端口如下，代码片段仅列出与接收有关的连接：

```
IDDR #(
    .DDR_CLK_EDGE ("SAME_EDGE_PIPELINED"),
    .INIT_Q1      (1'b0),
    .INIT_Q2      (1'b0),
    .SRTYPE       ("ASYNC")
) IDDR_adc_data (
    .Q1 (data_rise),
    .Q2 (data_fall),
    .C  (capture_clk),
    .CE (1'b1),
    .D  (data_delay),
    .R  (rst),
    .S  (1'b0)
);
```

接收顶层 ACFL3432_ctrl.v 使用 generate 循环复制 14 组单 bit 电路，并对两路 14 位总线各打一拍。训练器检查的就是这两路公共寄存器输出，采样通道选择也从相同位置取数。

```
genvar i;
generate
    for (i = 0; i < 14; i = i + 1) begin : GEN_ADC_DATA_CAPTURE
```

```
cm3432_dds_data_bit #(
    .IDELAY_VALUE(0)
) cm3432_dds_data_bit_inst (
    .capture_clk (adc1_clk),
    .rst         (rst),
    .data_p      (adc1_data_p[i]),
    .data_n      (adc1_data_n[i]),
    .tap_value   (train_tap_value),
    .tap_load    (train_tap_load),
    .data_rise   (adc1_data_a_iddr[i]),
    .data_fall   (adc1_data_b_iddr[i])
);

end
endgenerate

always @(posedge adc1_clk or posedge rst) begin
    if (rst) begin
        adc1_data_a_reg <= 14'd0;
        adc1_data_b_reg <= 14'd0;
    end
    else begin
        adc1_data_a_reg <= adc1_data_a_iddr;
        adc1_data_b_reg <= adc1_data_b_iddr;
    end
end
end
```

注意：IDELAYCTRL 与各 IDELAYE2 的 IODELAY_GROUP 均为 cm3432_idelay。调试时应同时确认 200 MHz 参考时钟、复位、RDY 和 tap 装载脉冲。

1.3.1. 3CM3432 上电与 SPI 配置

ADC 上电后不能立即开始采集。AD_ctrl_top_tcp 首先通过 cm3432_powerup 管理 PDWN，再允许 lut_config 执行寄存器初始化。该过程在 50 MHz 控制时钟域运行。

```
cm3432_powerup #(
    .CLK_HZ          (50_000_000),
    .PDWN_HIGH_US   (10),
    .POWERUP_SETTLE_US (300)
) cm3432_powerup_inst (
    .clk             (clk_50M),
    .rst_n           (reset_n),
    .adc_pdwn        (adc_pdwn),
    .init_ready      (adc_spi_init_ready)
);
```

表 1-5 上电控制过程

阶段	动作	进入下一阶段的条件
复位与保持	PDWN 为高，初始化允许保持为 0	复位释放后高电平保持约 10 μ s
唤醒等待	PDWN 拉低，ADC 开始唤醒	再等待约 300 μ s
配置允许	adc_spi_init_ready 拉高	解除 lut_config 复位并开始配置
逐项初始化	依次发送寄存器地址和数据	每项等待 SPI 完成后处理下一项
训练准备	0x000D=0x04，使 ADC 输出棋盘格	接收端在就绪条件和延时满足后开始扫描

init_ready 的含义是“上电等待已经完成，可以启动 SPI”，而不是“整张初始化表已经写完”。训练启动端使用这一允许信号和 IDELAY 就绪状态，再进行固定延时，给 SPI 配置和返回时钟留出稳定时间。

1.3.1.3.1 配置模块的分工

lut_config 保存初始化表及待执行的运行时命令；spi_ctrl 负责 SPI 事务控制和三线数据口方向。lut_index 对应一次 SPI 事务完成事件，驱动初始化索引递增或运行时状态切换。lut_data 为 24 位，由 16 位指令部分和 8 位数据组成。

1.3.1.3.2 CM3432 上电寄存器初始化表

lut_config.v 中 INIT_LAST=20，初始化索引为 0~20。系统启动时按照下表顺序写入，最后通过 0x00FF 提交影子寄存器配置。训练使用的测试模式也在这张表中设置。

表 1-6 上电 SPI 初始化顺序

索引	寄存器地址	写入值	配置作用
0	0x0000	0x3C	软件复位
1	0x0005	0x03	选择两个 ADC 通道
2	0x0008	0x00	正常电源工作模式
3	0x000B	0x00	默认时钟设置，不分频
4	0x000D	0x04	棋盘格测试模式，用于 tap 训练
5	0x0010	0x00	不进行偏移调整
6	0x0014	0x05	LVDS 输出及二进制补码格式
7	0x0015	0x01	LVDS 输出驱动配置

索引	寄存器地址	写入值	配置作用
8	0x0016	0x00	交织 LVDS 输出及正常 DCO 配置
9	0x0017	0x00	DCO 延时配置关闭
10	0x0018	0x0F	ADC 芯片内部模拟量程配置
11	0x0019	0x00	用户测试图案寄存器
12	0x001A	0x00	用户测试图案寄存器
13	0x001B	0x00	用户测试图案寄存器
14	0x001C	0x00	用户测试图案寄存器
15	0x001D	0x00	用户测试图案寄存器
16	0x001E	0x00	用户测试图案寄存器
17	0x001F	0x00	用户测试图案寄存器
18	0x0020	0x00	用户测试图案寄存器
19	0x003A	0x00	同步功能配置
20	0x00FF	0x01	提交影子寄存器

0x0018=0x0F 在初始化代码中标注为芯片内部 2.087 V_{pp} 输入跨度。这里的参数属于 ADC 芯片端，不能直接替代模块外部 ±5 V 的量程定义。改变模拟量程后，还需要重新确认上位机的电压标定。

1.3.1.3.3 运行时 SPI 访问与影子寄存器提交

PS 将寄存器访问编码为 32 位 reg_conf。bit24 为读写方向：0 表示写，1 表示读；实际 SPI 地址取 reg_conf[20:8]，即 13 位地址；低 8 位是写入数据。读操作的数据由 cm_readback 返回。

表 1-7 reg_conf 含义

字段	含义	示例
reg_conf[24]	读写方向	写 0x000D=0: 0x00000D00
reg_conf[20:8]	SPI 寄存器地址	读 0x000D: 0x01000D00
reg_conf[7:0]	写入值	读操作不把该字段当作读回数据

字段	含义	示例
cm_readback[7:0]	最近一次 SPI 读回值	由 AXI 偏移 0x1C 读取

本工程仅对地址 0x0008~0x0020 以及 0x003A 的写操作自动追加 0x00FF=0x01。读操作不追加提交；直接写 0x00FF 时也不会再次追加。相关判断如下：

```
function is_shadow_register;
    input [12:0] register_address;
    begin
        is_shadow_register =
            ((register_address >= 13'h008) && (register_address <= 13'h020)) ||
            (register_address == 13'h03A);
    end
endfunction
```

TRANSFER (GLO, 0xFF)

位	名称	访问类型	复位	描述
7:1	RSV	RW	0x00	保留位。
0	TRANSFER	RW	0	影子寄存器同步使能。该位使能后会将主寄存器的值更新至从寄存器，从而控制功能生效。更新完成后此位将自动清零。

图 1-8TRANSFER 寄存器与影子配置提交

SPI 时序采用 CPOL=0、CPHA=0。写事务由 FPGA 驱动 SPI_IO；读事务在指令阶段之后释放数据线，接收 ADC 返回的 8 位数据。读完成使用 spi_rd_done，包含提交动作的配置事务则需等待 spi_conf_done。

1.3.1.4 自动扫 tap 与训练图案检查

自动扫 tap 用于选择稳定的数字接收位置。训练期间，CM3432 输出已知的棋盘格图案，FPGA 逐个尝试输入延时值，并判断 A、B 两路数据是否正确。它调整的是接收数据的延时位置，不改变 ADC 的模拟采样频率。

1.3.1.4.1 启动条件

ACFL3432_ctrl 在 DCO 域检查 idelay_rdy 和同步后的 spi_init_ready。条件满足后启动 22 位计数器，计数达到全 1 时拉高 train_start。在默认 250 MHz 返回时钟下，等待时间约为 16.78 ms。

```
else begin
    spi_init_ready_sync <= {spi_init_ready_sync[0], spi_init_ready};
    retrain_toggle_sync <= {retrain_toggle_sync[1:0], retrain_toggle};
    if (!idelay_rdy || !spi_init_ready_sync[1]) begin
        train_delay_cnt    <= 22'd0;
        train_start        <= 1'b0;
        restart_low_count <= 3'd0;
        restart_active     <= 1'b0;
    end
    else if (retrain_event) begin
```

1.3.1.4.2 检查棋盘格图案

cm3432_tap_trainer 同时检查 A、B 两路是否属于 14'h1555 或 14'h2AAA，并检查每一路相邻有效样点是否交替。某一路出现不允许的图案，或连续两次出现相同图案，都会形成错误。

```

wire data_a_is_pattern = (data_a == 14'h2aaa) || (data_a == 14'h1555);
wire data_b_is_pattern = (data_b == 14'h2aaa) || (data_b == 14'h1555);
wire sample_error_next =
    !data_a_is_pattern ||
    !data_b_is_pattern ||
    (prev_a_valid && (data_a == prev_a)) ||
    (prev_b_valid && (data_b == prev_b));
    
```

该条件只要求 A、B 各自满足图案与交替关系，不要求同一时刻两路必须相等或必须相反。14 位总线共用 tap，因此任何数据位引起的整字错误都会影响公共窗口的判定。

1.3.1.4.3 自动扫描状态机

idelay_tap_window_trainer 包含七个状态。扫描从 tap=0 开始，依次尝试到 tap=31；每个 tap 先等待 64 个有效周期，再推进 1024 周期的检测窗口，容许错误计数 MAX_ERRORS=0。

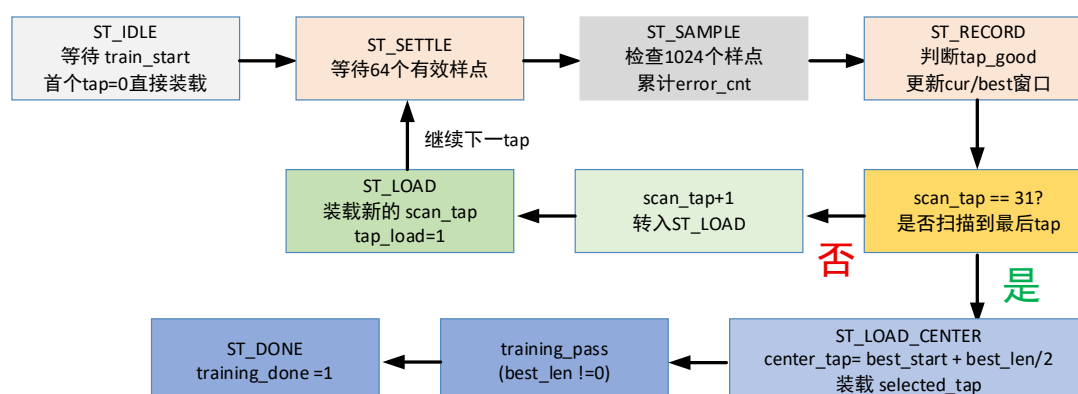


图 1-9 IDelay tap 自动扫描状态机

表 1-8 扫描状态及作用

状态	执行动作
ST_IDLE	等待启动，清空窗口记录，直接装载第一个 tap=0
ST_LOAD	装载后续 scan_tap，清零当前等待与错误计数
ST_SETTLE	等待延时单元和接收流水稳定，计满 64 个有效周期
ST_SAMPLE	推进检测窗口，遇到 sample_error 时累计错误
ST_RECORD	判断本 tap，更新当前连续窗口及最长窗口

状态	执行动作
ST_LOAD_CENTER	计算并装载最长窗口中心，更新 training_pass
ST_DONE	training_done 置 1，等待 train_start 撤销

tap_load 是一次装载脉冲，tap_value 是送给 IDELAY 的设置值，selected_tap 则保存最终选择结果。扫描中要观察 tap_value 的变化；上位机查询得到的是 selected_tap，不是每个扫描点的错误统计。

1.3.1.4.4 最长有效窗口与中心 tap 选择

在 idelay_tap_window_trainer.v 中，连续通过的 tap 构成一个有效窗口。cur_start、cur_len 记录正在延长的窗口；best_start、best_len 保存已找到的最长窗口。只有新的窗口长度严格大于 best_len 时才替换记录，因此等长窗口保留先扫描到的一个。

```

wire error_cnt_saturated = (error_cnt == {COUNT_WIDTH{1'b1}});
wire tap_good = (error_cnt <= MAX_ERRORS);
wire [TAP_BITS:0] cur_len_next = cur_len + 1'b1;
wire [TAP_BITS:0] best_center_sum = {1'b0, best_start} + (best_len >> 1);
wire [TAP_BITS-1:0] center_tap =
    (best_len == 0) ? START_TAP_VALUE : best_center_sum[TAP_BITS-1:0];
.....
    ST_LOAD_CENTER: begin
        selected_tap <= center_tap;
        tap_value    <= center_tap;
        tap_load     <= 1'b1;
        training_pass <= (best_len != 0);
        state        <= ST_DONE;
    end

```

15

表 1-9 窗口中心计算示例

算法示例	结果	说明
tap 8~15 通过	选择 tap 12	起点 8 + 窗口长度 8/2
tap 3~6、20~23 通过	选择 tap 5	两段等长，保留先遇到的窗口
仅 tap 9 通过	pass=1, tap=9	通过条件为存在至少一个有效 tap
所有 tap 均失败	pass=0, tap=0	回退值使用 START_TAP=0
tap 28~31 与 0~2 通过	按两段窗口处理	扫描不进行首尾环形合并

以上数值用于说明算法，不是固定的板级训练结果。不同板卡、时钟设置和接收条件下，selected_tap 可以不同；每次使用都应检查 done 与 pass，而不是只与某一个 tap 数值比较。

注意：检测错误先经过一级寄存器。调整训练参数或修改判定逻辑时，需要在仿真中覆盖窗口首尾，特别是最后一个样点单独出错的情况。1024 是窗口计数设置，不是长期误码率或时序裕量的保证。

1.3.1.4.5 训练完成、退出测试模式与采集门控

上电初始化将 ADC 留在棋盘格模式。扫描结束后，AD_ctrl_top_tcp.v 把 training_done 同步到 50 MHz 控制域，并检测完成上升沿，自动请求写 0x00D=0x00，使 ADC 恢复正常模拟转换数据输出。

```
wire training_done_rise = training_done_sync[1] & ~training_done_sync[2];
....
always @(posedge clk_50M or negedge reset_n) begin
    if (!reset_n) begin
        adc_normal_conf_active <= 1'b0;
        adc_data_ready          <= 1'b0;
    end else if (!adc_spi_init_ready) begin
        adc_normal_conf_active <= 1'b0;
        adc_data_ready          <= 1'b0;
    end else begin
        if (training_done_rise && !adc_data_ready)
            adc_normal_conf_active <= 1'b1;

        if (adc_normal_conf_active && spi_conf_done) begin
            adc_normal_conf_active <= 1'b0;
            adc_data_ready          <= 1'b1;
        end
    end
end
end
```

16

这次写操作经 SPI 配置路径执行，并等待影子寄存器提交完成，随后置 adc_data_ready。后续数据有效还需要训练通过，完整门控如下：

```
wire capture_data_ready = adc_data_ready && training_pass_sync[1] &&
                           !runtime_retrain_busy;
wire sample_enable = ad_sample_en_sync && capture_data_ready;
....
assign ad_out_valid = converted_valid && adc_data_en;
```

表 1-10 扫描状态

状态	含义	使用要求
TRAIN_DONE	扫描已经结束	不能单独代表找到可用 tap
TRAIN_PASS	至少发现一个有效 tap	采集前必须为 1
DATA_READY	上电训练后的退出测试模式事务完成	与 TRAIN_PASS 一起判断

状态	含义	使用要求
runtime_retrain_busy	正在执行运行时重训练	为 1 时关闭采集有效数据

扫描没有通过时，控制逻辑仍可能完成退出棋盘格，因此 DATA_READY=1 并不等于数据可采集。PS 启动采样前同样检查 DATA_READY 与 TRAIN_PASS。调试时不要绕过训练通过条件来取得表面上“有输出”的波形。

1.3.1.5 修改 0x000B 后的自动重训练

1.3.1.5.1 触发条件

运行时改变 ADC 时钟配置后，数据与 DCO 的关系可能发生变化。

cm3432_runtime_retrain.v 将这类写请求组织成完整事务，在应用新配置后重新寻找 tap，再恢复正常数据模式。

```

wire host_is_write = !host_cmd_conf[24];
wire host_is_clk_setting =
    host_is_write && (host_cmd_conf[23:8] == 16'h000B);
wire host_clk_setting_changed =
    !trained_setting_valid ||
    (host_cmd_conf[5:0] != trained_clk_setting);

assign intercept_host_cmd =
    (state == ST_IDLE) && host_cmd_valid && host_is_clk_setting &&
    host_clk_setting_changed;

```

17

只有写 0x000B，且低 6 位与已成功训练记录不同，或尚无有效记录时，才进入自动重训练。读取该寄存器不会触发；对已成功训练的同一低 6 位重复写入，走普通 SPI 写路径。

表 1-11 自动重训练事务顺序

顺序	操作	完成条件
1	写 0x000D=0x04，进入棋盘格	等待 SPI 事务及提交完成
2	写 0x000B=新配置值	等待 SPI 事务及提交完成
3	等待 DCO 稳定	50 MHz 域计时约 10 μ s
4	翻转 retrain_toggle	接收域重新触发扫描
5	等待新的训练结果	先等旧 done 变 0，再等新 done 变 1
6	写 0x000D=0x00，恢复正常模式	等待提交后返回本次命令结果

1.3.1.5.2 重训练状态与完成握手

重训练控制器使用十个状态，SPI 操作采用“发出一次请求，再等待完成”的形式。这样可以保证测试模式、新分频配置和正常模式切换按顺序生效。

表 1-12 重训练状态

状态	主要任务
ST_IDLE	识别需要重训练的寄存器写请求
ST_TEST_ISSUE / ST_TEST_WAIT	发出棋盘格配置，等待完成
ST_DIV_ISSUE / ST_DIV_WAIT	发出 0x000B 新值，等待完成
ST_DCO_SETTLE	等待 10 μ s，发出重训练翻转事件
ST_WAIT_TRAIN_LOW	确认旧的 training_done 已经撤销
ST_WAIT_TRAIN_END	等待扫描完成，锁存本次 pass
ST_NORMAL_ISSUE / ST_NORMAL_WAIT	退出棋盘格，返回命令完成及成功状态

接收域检测到 retrain_toggle 变化后，先把 train_start 拉低数个 DCO 周期，让停在 ST_DONE 的扫描器回到 ST_IDLE，再重新拉高开始扫描。运行时重训练不重复上电的 22 位长延时。

18

```
ST_NORMAL_WAIT: begin
    if (spi_cmd_done) begin
        if (retrain_pass_latched) begin
            trained_clk_setting    <= pending_clk_setting[5:0];
            trained_setting_valid <= 1'b1;
        end
        busy                <= 1'b0;
        host_cmd_done       <= 1'b1;
        host_cmd_success    <= retrain_pass_latched;
        state               <= ST_IDLE;
    end
end
```

无论扫描是否通过，控制器都会尝试恢复正常模式；只有 pass 为 1 且恢复事务结束，才返回成功。上位机应等待写应答，再查询 tap 和测试模式寄存器。

注意：0x0016、0x0017 等其他时序相关寄存器没有同样的自动拦截流程。PS 的约 2 秒命令超时只结束软件等待，不会自动复位 PL 重训练状态机；持续忙时应检查 DCO、SPI 完成事件和状态机停留位置。

1.3.1.6 通道选择与 16 位存储格式

ad_14bit_to_16bit.v 从测试数据、A 路和 B 路中选择一个来源，形成一路 16 位输出。ADC 有效数据保留在低 14 位，高两位补 0。这里是存储位宽扩展，符号仍需按有效 14 位解释。

```
case (ch_sel)
    2'b00: ad_out <= {2'd0, adc_test_data};
    2'b01: ad_out <= {2'd0, ad_in1};
    2'b10: ad_out <= {2'd0, ad_in2};
    default: ad_out <= 16'd0;
endcase
...
always @(posedge clk or negedge rst_n) begin
    if (!rst_n)
        ad_out_valid <= 1'b0;
    else
        ad_out_valid <= ad_sample_en && (ch_sel != 2'b11);
end
```

表 1-13 通道选择

CH_SEL	数据来源	每个样点的传输长度
0	FPGA 内部 14 位递增计数	2 字节
1	ADC 通道 1（A 路）	2 字节
2	ADC 通道 2（B 路）	2 字节
3	禁用，不进行双路组合	无有效样点

19

1.3.1.7 采样速率与时间轴设置

ADC 的实际数据更新速度由采样时钟和芯片时钟配置决定。本工程 speed_ctrl.v 保留 div_set 接口，但 adc_data_en 固定为 1，采集数据不进行计数抽取。

```
module speed_ctrl(
    input wire      clk,
    input wire      reset_n,
    input wire      ad_sample_en,
    output wire      adc_data_en,
    input wire [31:0] div_set
);
assign adc_data_en = 1'b1;
```

实际改变采样速率，通过修改 0x0B 寄存器的低 3 位实现 1~8 分频。

CLK DIVIDE (GLO, 0x0B)

位	名称	访问类型	复位	描述
7:6	RSV	RW	00	保留位。
5:3	CLK PHASE ADJUST	RW	000	时钟分频相位调整。 000: 无延迟 001: 1 个输入时钟周期 010: 2 个输入时钟周期 011: 3 个输入时钟周期 100: 4 个输入时钟周期 101: 5 个输入时钟周期 110: 6 个输入时钟周期 111: 7 个输入时钟周期
2:0	CLK DIVIDE RTO	RW	000	时钟分频调整。 000: 1 分频 001: 2 分频 010: 3 分频 011: 4 分频 100: 5 分频 101: 6 分频 110: 7 分频 111: 8 分频

图 1-10 分频寄存器

TCP 命令 0x03 可以设置 Speed_Set, PS 也会将它写入 SPEED_CTRL 寄存器, 但它不改变当前采样选通频率。上位机不能仅修改这一字段, 就按较低的采样率解释接收到的数据。

表 1-14TCP 命令 0x03 可以设置

20

设置对象	作用	使用方法
Clocking Wizard 默认输出	提供 250 MHz ADC 时钟和 200 MHz 延时参考	本章首先按默认时钟配置验证
CM3432 的 0x000B	改变 ADC 时钟相关配置	有效低 6 位变化可触发自动重训练
Speed_Set / 命令 0x03	保留速度字段, 不执行抽取	默认测试设置为 0
PC 采样率与时间间隔	决定绘图横轴和频率测量	按本轮真实采样条件填写

默认 250 MSPS 对应每个样点 4 ns。采集 N 个样点时, 时间坐标为 $t[n]=n/F_s$ 。修改 0x000B 后, 设备信息结构体中的 250000000 和 4 ns 不会自动更新, 应同步设置上位机横轴。

注意: 命令 0x05 的 Clocking Wizard 重配置入口未与自动训练、参考时钟和上位机参数建立完整联动。本章操作流程保持默认 PLL 配置, 不使用该入口完成在线调频。0x000B 的具体分频编码应按所用 CM3432 芯片寄存器定义选择。

1.3.1.8 异步 FIFO 与数据流格式转换

fifo2axis_write_ADC 接收一路 16 位样点，通过异步 FIFO 将 ADC 返回时钟域的数据送入 AXIS 时钟域。FIFO 写入位宽为 16 位，读出位宽为 128 位，每次读出包含 8 个样点。

表 1-15 异步 FIFO 参数

配置项	参数
存储实现	Independent Clocks Block RAM
写入位宽 / 深度	16 位 / 131072 字
读出位宽 / 深度	128 位 / 16384 字
名义存储容量	256 KiB
读取方式	Standard FIFO，读请求后一拍返回数据
写时钟	M_AXIS_FIFO_CLK，即 ADC 返回时钟
读时钟	M_AXIS_ACLK，即 100 MHz AXIS 时钟

1.3.1.8.1 写入计数与使能

模块将 Go、ad_sample_en 同步到写时钟域；启动一段采集时，将 ByteToTrans/2 作为本段需写入的样点数量。输入 din 和 dvalid 先经过一级寄存，再进行 FIFO 写入。

```
assign wr_fifo = dvalid_pipe && write_active && mode_valid &&
                !fifo_full && !fifo_wr_busy;
....
samples_remaining <= ByteToTrans[25:1];
....
if (wr_fifo) begin
    samples_remaining <= samples_remaining - 1'b1;
    if (samples_remaining == 25'd1)
        write_active <= 0;
end
```

计数只在实际写入成功时递减，写满本段规定数量后停止。FIFO 的复位由局部控制状态机完成，不需要每轮采样都重新复位 ADC 配置与训练链路。

注意：FIFO 满时禁止写入，并置 fifo_overflow。ADC 不会因 FIFO 满而停止转换，因此该时刻的样点会丢失；即使后来累计字节数正确，也仍需检查是否出现采样缺口。

1.3.1.8.2 128 位 AXIS 握手、打包与 TLAST

FIFO 采用普通读模式，读请求与返回数据之间有一拍延迟。模块设置 beat0、beat1 两个 128 位暂存寄存器，并用 fifo_read_pending 跟踪在途读取，保证 AXIS 反压期间待发送数据保持稳定。

```
wire [21:0] beat_goal = ByteToTrans[25:4];
assign M_AXIS_TDATA = beat0;
assign M_AXIS_TVALID = (mst_exec_state == SEND_STREAM) && en_send &&
                        mode_valid && (buffered_beats != 0);
assign M_AXIS_TLAST = M_AXIS_TVALID && (beats_sent == beat_goal - 1'b1);
assign axis_fire = M_AXIS_TVALID && M_AXIS_TREADY;
```

一次有效传输必须同时满足 TVALID 和 TREADY，只有 axis_fire 为 1 才增加 beats_sent。TLAST 标识本段最后一个 128 位数据拍；若最后一拍遇到反压，数据和 TLAST 仍需保持到握手完成。

1.3.1.8.3 样点顺序与字节顺序

```
wire [127:0] reorder_dout =
    {dout_fifo[15:0], dout_fifo[31:16], dout_fifo[47:32],
     dout_fifo[63:48], dout_fifo[79:64], dout_fifo[95:80],
     dout_fifo[111:96], dout_fifo[127:112]};
```

reorder_dout 将 8 个 16 位片段重新排列，只改变样点在 128 位字内的位置，不交换单个样点内部的高低字节。PS 直接发送 DDR 内存字节，上位机按小端 16 位容器还原后，再进行 14 位补码解释。

1.3.1.8.4 长度对齐与预存量

一拍为 16 字节，beat_goal 直接取 ByteToTrans[25:4]，所以采集字节数必须是 16 的整数倍，即样点数为 8 的整数倍。读取启动逻辑使用 rd_data_count_fifo ≥ 64 的预存条件，并带有特殊分支：首次联调使用 4096 点、8192 字节，不从小于 1024 字节的短帧开始。短帧需要单独验证启动门槛。

1.3.1.9 AXI DMA 数据搬运

DMA 将 FIFO 输出的 AXI4-Stream 数据写入 PS DDR。本实验仅使用 S2MM 写入通道，不启用 MM2S，也不使用 Scatter-Gather。数据流宽度和 DMA 内存侧宽度均为 128 位，经 AXI 互连适配后连接 PS 的 64 位 HP0 接口。

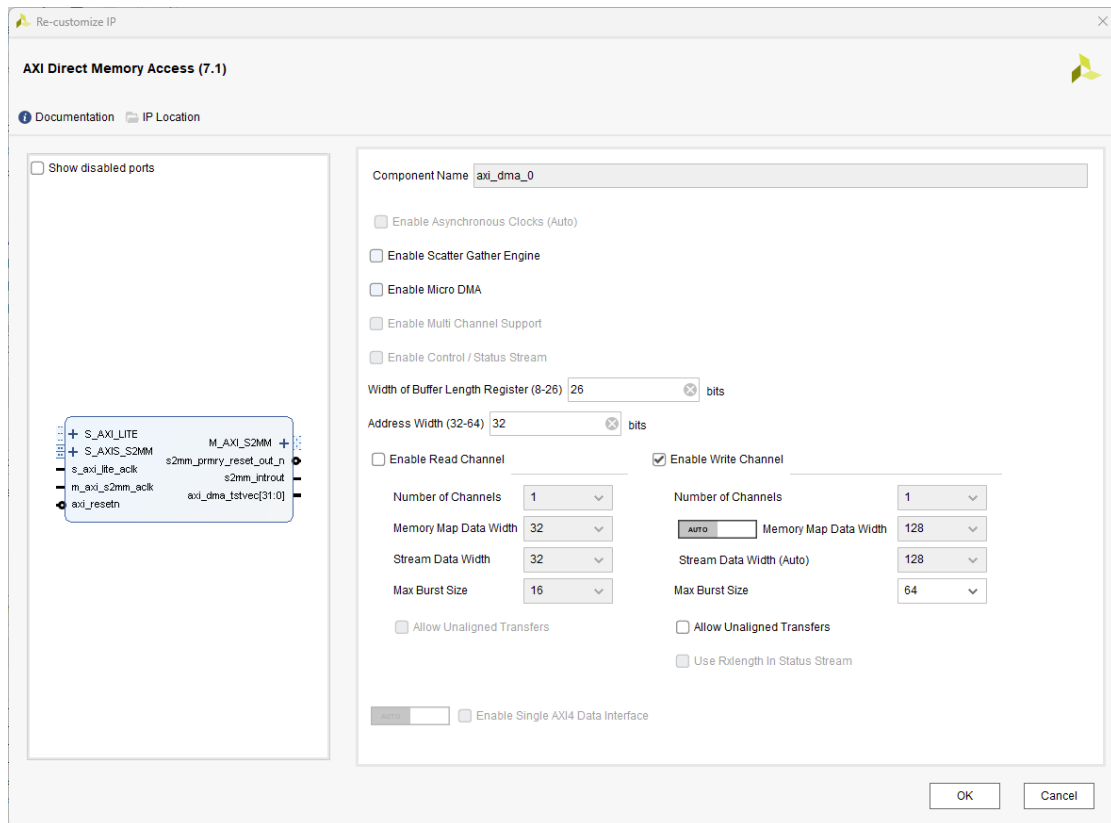


图 1-11DMA IP 配置

表 1-16DMA 配置与缓存参数

配置或常量	取值	含义
S2MM Stream Data Width	128 位	每拍承载 8 个 16 位样点
S2MM Memory Map Data Width	128 位	DMA 内存侧宽度
PS S_AXI_HP0	64 位	DDR 高性能访问接口
Buffer Length Register Width	26 位	DMA 长度字段宽度
DMA_CHUNK_MAX	0x03FF0000	每段最多 67,043,328 字节
CAPTURE_MAX_BYTES	0x3E800000	单轮软件上限 1,048,576,000 字节
ADC_Data_Addr	0x01800000	DDR 采集起始地址

上位机设置的总字节数超过单段上限时，PS 分多次提交 DMA。每一段使用自己的 ByteToTrans 和 TLAST，完成后把 DDR 写入地址向后移动，再开始下一段。所有段采集结束后才发送 TCP 数据。

1.3.1.9.1 分段计算示例

请求 80 MiB，即 83,886,080 字节时，分为两段：第一段 67,043,328 字节，第二段 16,842,752 字节。两段长度均满足 16 字节对齐，DDR 中的数据首尾相接。

注意：DDR 地址连续不等于采样时间连续。软件在段间处理完成状态并重新启动 DMA，可能产生采样间隔。本工程用于有限长度记录，不是持续环形 DMA 与网络同时运行的无间断采集系统。

执行接近容量上限的测试前，应检查链接 map、程序与堆栈以及其他 DDR 缓冲区，确保采集区域没有重叠。软件长度上限只检查地址范围，不替代完整的内存区域规划。

1.3.1.9.2 指令接收转控制模块

上位机下发的指令需要进行解析，这里使用自定义 IP `adc_sample_ctrl` 模块来解析指令，解析完成的指令给到其他模块使用。

1. 将 `slv_reg0` 的 bit0 位给到 `ad_sample_en` 信号，作为 ADC 采样使能信号；
2. 将 `slv_reg0` 的 bit8 位给到 `Go` 信号，作为 DMA 传输使能信号；
3. 将 `slv_reg1` 的 bit1 和 bit0 两位给到 `ChannelSel` 信号，作为通道选择信号；
4. 将 `slv_reg2` 给到 `ADC_Speed_Set` 信号，作为速度设置；
5. 将 `slv_reg3` 的低 26 位给到 `ByteToTrans` 信号，作为一轮 DMA 传输字节数量；
6. 将 `slv_reg4` 给到 `Data_Sum` 信号，作为采样总字节数；
7. 将 `slv_reg5` 给到 `reg_conf` 信号，作为需要修改的具体寄存器和具体值；

```
assign ad_sample_en    = slv_reg0[0];  
assign Go              = slv_reg0[8];  
assign ChannelSel      = slv_reg1[1:0];  
assign ADC_Speed_Set   = slv_reg2;  
assign ByteToTrans     = slv_reg3[25:0];  
assign Data_Sum        = slv_reg4;  
assign reg_conf        = slv_reg5;
```

24

1.3.1.10AXI 控制寄存器

自定义 `adc_sample_ctrl_0` IP 为 PS 提供寄存器接口。TCP 命令由 PS 解析，PL 根据寄存器输出完成采集和 ADC 配置。工程导出的控制基地址为 0x43C10000，应用通过 `xparameters.h` 及相应宏访问。

表 1-17 控制寄存器映射

偏移	名称	用途
0x00	SAMPLE_EN_GO	bit0: 采样使能; bit8: Go
0x04	CH_SEL	有效低 2 位, 0/1/2 选择测试或单通道
0x08	SPEED_CTRL	速度接口参数, 当前不执行抽取

偏移	名称	用途
0x0C	BYTES_Trans	本段 DMA 字节数，低 26 位有效
0x10	DATA_SUM	本轮总采集字节数
0x14	REG_CONF	ADC 寄存器读写命令，每次写入触发请求
0x18	CM3432_STATUS	读取 ADC 训练与命令状态
0x1C	CM3432_READBACK	最近一次 SPI 读回值，低 8 位有效

1.3.1.10.1 采集控制值

表 1-18 采集控制值

写入值	ad_sample_en	Go	使用含义
0x00000000	0	0	停止采集与传输控制
0x00000100	0	1	准备阶段关闭采样，Go 保持有效
0x00000101	1	1	开放采样和本段数据传输

1.3.1.10.2 重复寄存器命令

每次 AXI 写入 REG_CONF，reg_cmd_toggle 都翻转一次。控制域通过检测翻转事件接收新命令，因此连续两次写入完全相同的命令字也会执行两次。完成时 cmd_done_toggle 再次翻转，PS 比较请求前后值判断完成，而不是等待一个固定的高电平完成标志。

注意：0x18 和 0x1C 的读取结果来自外部状态与读回信号，不能通过写这两个地址来设置 tap、清训练状态或修改 SPI 读回值。

1.3.1.10.3 CM3432 状态字与跨时钟域连接

CM3432_STATUS 高 16 位固定为 0x3432，低位包含训练、配置和命令状态。软件先识别签名，再解释具体字段。

表 1-19 CM3432_STATUS 位定义

位段	状态名	含义
31:16	签名	固定为 0x3432
15:13	保留	0
12:8	selected_tap	最终选中的 5 位 tap
7	TRAIN_PASS	扫描发现有效 tap

位段	状态名	含义
6	TRAIN_DONE	扫描结束
5	DATA_READY	上电退出棋盘格事务完成
4	INIT_READY	PDWN 唤醒等待结束，允许 SPI 初始化
3	CMD_FAILED	最近一次硬件命令失败
2	DONE_TOGGLE	每次命令完成翻转
1	BUSY	直接 SPI 访问或运行时重训练正在执行
0	READY	当前允许寄存器访问

READY 是寄存器访问的准备条件，TRAIN_PASS 是采样有效条件，两者不能互相代替。上电期间 BUSY 也不直接包含整个扫描过程，应综合 DATA_READY、TRAIN_DONE 与 TRAIN_PASS 判断。

1.3.1.10.4 跨时钟域处理

AXI 请求通过翻转事件送入 50 MHz 控制域；训练结果从 DCO 域同步至控制域；采样数据通过异步 FIFO 跨入 AXIS 域。状态字及 SPI 读回数据进入 AXI 域前经过两级寄存器同步。

寄存器访问采用“提交一条命令、等待完成、读取结果”的顺序，使命令字和结果在跨域期间保持稳定。上位机查询最终 tap 时也应等待训练完成，不把扫描过程中一次读到的多位状态当作严格同周期的快照。

1.3.1.11 Zynq IP 与 Block Design 配置

PS 侧配置千兆以太网、串口、DDR、控制接口和 DMA 数据接口。工程器件为 xc7z020clg484-2。PS Bank 电压由硬件决定，Bank0 配置为 LVCMOS 3.3 V，Bank1 配置为 LVCMOS 1.8 V。

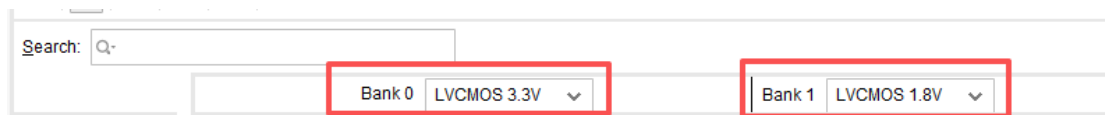


图 1-12 PS Bank 电压设置

1.3.1.11.1 使能 PS 端千兆以太网

勾选 ENET0，使用 MIO16~27；同时使能 MDIO，使用 MIO52~53。TCP 命令和采样数据均经 PS 侧网口传输，板级连接时应选择对应的 PS_ETH 接口。

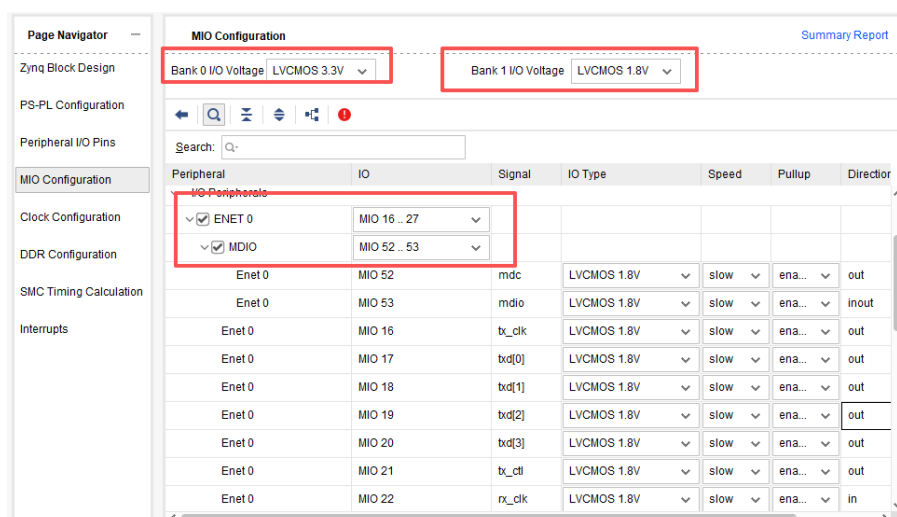


图 1-13ENET0 与 MDIO 引脚设置

1.3.1.11.2 使能 PS 串口

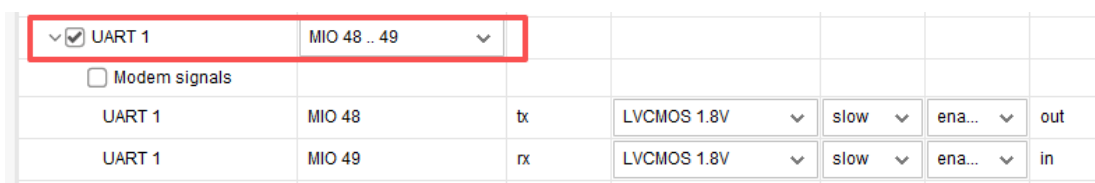


图 1-14UART1 引脚设置

1.3.1.11.3 AXI 接口、存储与时钟配置

M_AXI_GP0 用于 PS 访问自定义控制寄存器、DMA 控制寄存器和 Clocking Wizard；S_AXI_HP0 用于 DMA 向 DDR 高速写入数据。S2MM 完成中断通过 s2mm_introut 连接到 IRQ_F2P。

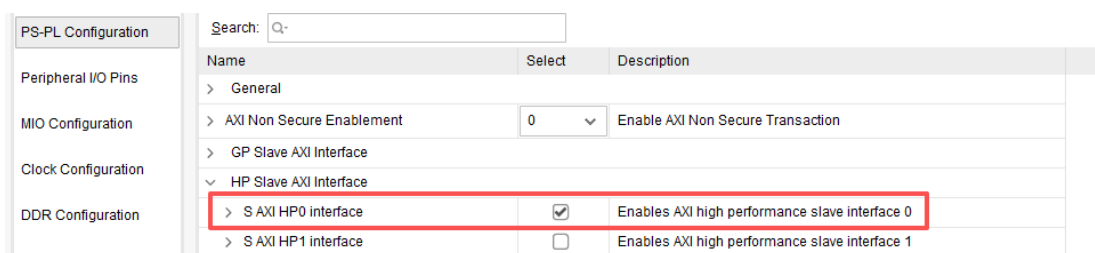


图 1-15 使能 PS 的 S_AXI_HP0 接口

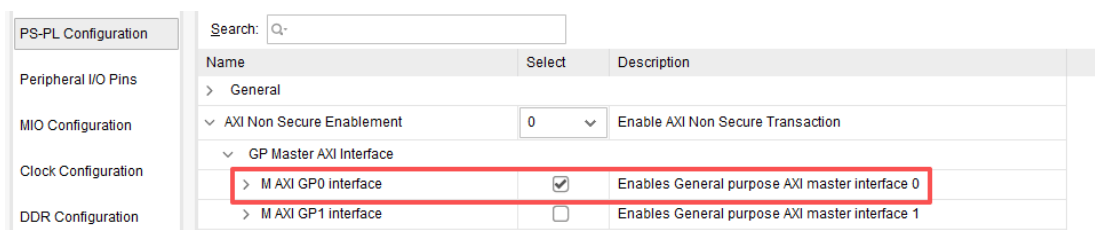


图 1-16 使能 PS 的 M_AXI_GP0 接口

表 1-20zynq IP 设置表

配置项	工程设置
DDR 类型与位宽	DDR3, 32 位
DDR 器件参数	MT41K256M16 RE-125
DDR 时钟	约 533.333 MHz
FCLK_CLK0	100 MHz, 驱动 AXI 与 DMA 相关逻辑
Clocking Wizard 输出	clk_out1=50 MHz; clk_out2=250 MHz; clk_out3=200 MHz
QSPI	MIO1~6, Single SS, 4-bit
SD0	MIO40~45
中断	使能 PL-PS 中断, 连接 DMA S2MM 中断

在 Block Design 中检查各时钟的实际端口连接: 50 MHz 送入 ADC 控制, 250 MHz 送入采样时钟转发, 200 MHz 送入 IDELAYCTRL, 100 MHz 连接 FIFO 读侧和 AXIS。完成连接后执行设计检查, 再生成 HDL Wrapper 与硬件输出文件。

1.3.1. 12AC820 采集接口引脚约束

下面列出本工程使用的关键正端和单端信号, 负端应与对应的封装差分对配套。

表 1-21AC820 关键管脚

端口	PACKAGE_PIN	电平标准
reset_n_0	H18	LVC MOS25
adc1_clk_p_0 (DCO)	L18	LVDS_25
adc_clk_p_0 (输出时钟)	J20	LVDS_25
adc1_spi_ce_0	L21	LVC MOS25
adc1_spi_sclk_0	B19	LVC MOS25
adc1_spi_io_0	D22	LVC MOS25
adc1_pdwn_0	P16	LVC MOS25
adc1_data_p_0[13:10]	G15 / A18 / A16 / D16	LVDS_25
adc1_data_p_0[9:6]	A21 / B21 / F21 / J16	LVDS_25

端口	PACKAGE_PIN	电平标准
adc1_data_p_0[5:2]	J18 / K19 / K16 / P17	LVDS_25
adc1_data_p_0[1:0]	J15 / N17	LVDS_25

```
set_property PACKAGE_PIN H18 [get_ports reset_n_0]
set_property IOSTANDARD LVCMOS25 [get_ports reset_n_0]
set_property PACKAGE_PIN L18 [get_ports adc1_clk_p_0]
set_property PACKAGE_PIN J20 [get_ports adc_clk_p_0]
set_property PACKAGE_PIN P16 [get_ports adc1_pdown_0]
```

返回 DCO 按默认 4 ns 周期建立时钟，并与 PS/MMCM 时钟组进行异步关系处理。自动训练解决运行时 tap 的选择，但不替代输入时序约束和跨时钟域检查。实现后仍需检查时序报告、时钟路径和 DRC。

注意：本节管脚对应 AC820 工程。更换主板时，应重新核对器件封装、FMC 插座、差分极性、PDWN、电平标准和 Bank 供电，不可直接沿用本表下载到其他硬件。

1.3.2 PS 端应用程序设计

1.3.2.1 软件工程搭建

PS 应用采用 Standalone 与 lwIP RAW API，不依赖 Linux。完成 Vivado 硬件设计后，生成 bitstream 并导出硬件，导出时包含 bitstream，再在 SDK 中使用匹配的硬件平台构建 APP 和 APP_bsp。

在创建 platform project 时，在 Board Support Package，需要勾选 lwip202 选项，如下图所示，xilffs 选项是固化操作需要的，大家根据自己需求是否勾选。

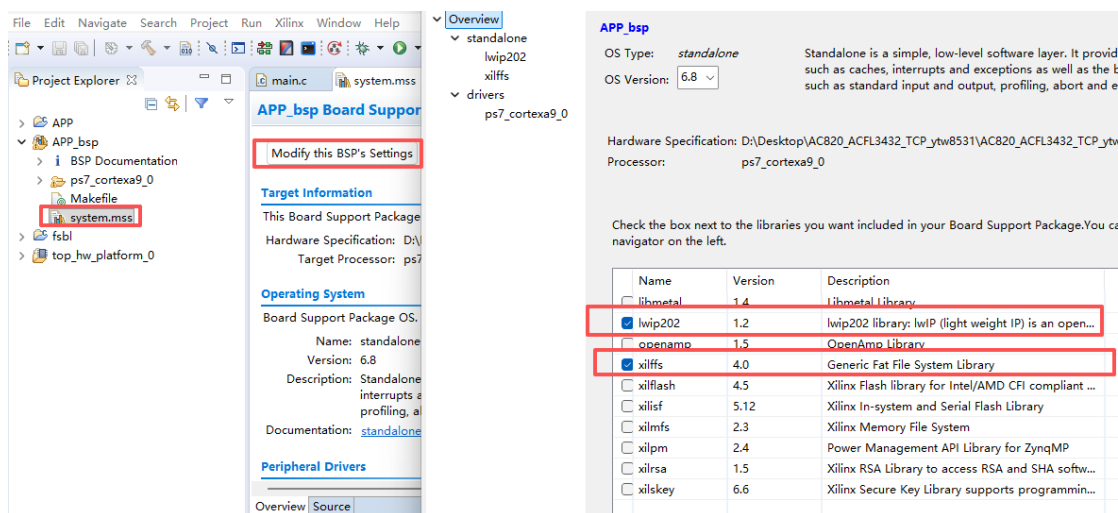


图 1-17platform 勾选 lwip202 选项

创建应用可从 lwIP Echo Server 模板开始，随后使用本工程 APP/src 中的应用文件。BSP 中应包含 lwip202，并保持与导出硬件一致的外设地址、中断和时钟配置。

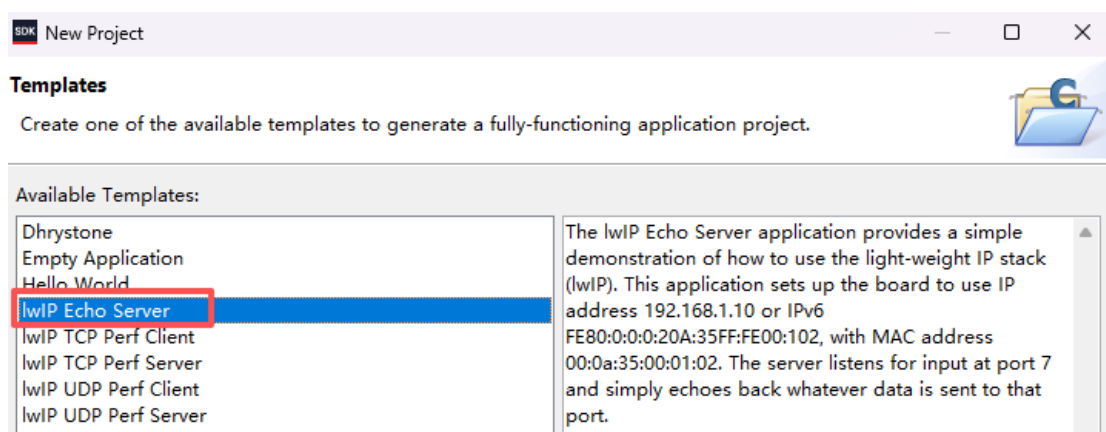


图 1-18 选择 lwIP Echo Server 应用模板

表 1-22lwIP Echo Server 应用模板文件

文件	作用
main.c	平台与网络初始化、主循环任务调度
echo.c	建立连接、接收 TCP 字节流、命令组帧
CMD_Handle.c / CMD_Handle.h	控制协议、寄存器服务、采样启动和数据发送
ADC_Info.h	CM3432 设备信息结构体
platform_zynq.c	定时器、GIC、DMA 初始化和中断处理
lscript.ld	应用内存布局和堆栈分配

30

实际编译目录为工程 SDK 工作区的 APP/src，构建时使用 APP 对应的 BSP。工程中还存在 SDK 根目录 src 副本，编译与修改应始终以 APP 的 Sources 和构建规则所指向的文件为准。

固化时使用同一硬件平台生成的 FSBL、bitstream 和 APP ELF 创建启动镜像。涉及 SD/FAT 文件访问的工程按相应配置启用 xilffs，并确保 BOOT 镜像中的各项与正在使用的硬件匹配。

为了实现设计目标，需要对其中的 echo.c、echo.h、main.c 等多个文件内容进行修改，由于涉及到的内容较多，用户可以直接将例程中这 16 个文件拷贝替换当前新建的工程文件。

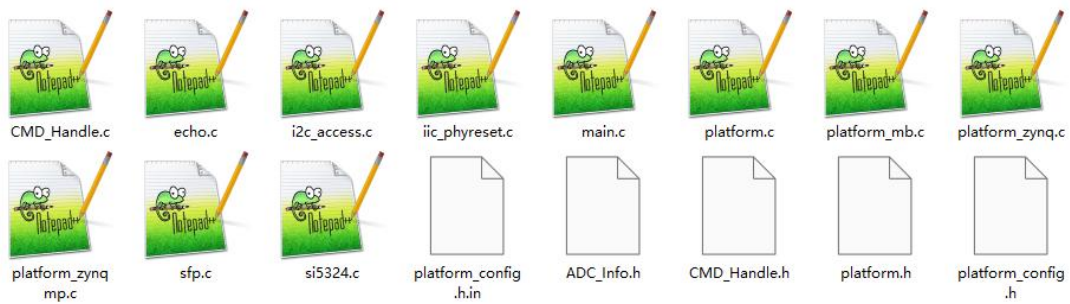


图 1-19 16 个拷贝替换文件

1.3.2.2 lwIP 与平台初始化

init_platform() 依次设置私有定时器、中断控制器和 AXI DMA。私有定时器为 TCP 定时处理提供节拍，GIC 统一注册定时器与 DMA 中断，避免各模块分别初始化中断控制器时互相覆盖。

```
void init_platform()
{
    platform_setup_timer();
    platform_setup_interrupts();
    platform_setup_axidma();
    return;
}
```

表 1-23 lwIP 与平台初始化函数说明 ‘

初始化函数	作用
platform_setup_timer()	配置私有定时器及装载值
platform_setup_interrupts()	配置 GIC，注册异常、定时器和 DMA 接收中断
platform_setup_axidma()	初始化 DMA 实例与中断相关参数
lwip_init()	初始化 lwIP 协议栈
xemac_add()	添加 PS 以太网接口

31

1.3.2.3 网络地址

BSP 启用 DHCP，程序先尝试获取网络地址。DHCP 超时且未取得地址时，回退为 192.168.0.2，掩码 255.255.255.0，网关 192.168.0.1。连接带 DHCP 的网络时，应以串口打印的 Board IP 为准。

```
IP4_ADDR(&(echo_netif->ip_addr), 192, 168, 0, 2);
IP4_ADDR(&(echo_netif->netmask), 255, 255, 255, 0);
IP4_ADDR(&(echo_netif->gw), 192, 168, 0, 1);
```

MAC 地址为 00:0A:35:01:FE:C0，TCP 监听端口为 5000。若关闭 LWIP_DHCP，需同时检查 main.c 的非 DHCP 分支：该分支使用 192.168.1.10 和网关 192.168.1.1。

注意：重新生成 BSP 后，需检查板上实际 PHY 的初始化与 MDIO 访问。工程文件夹名称不能代替 PHY ID 和驱动分支检查，串口中的协商结果应与实际网口速率一起确认。

1.3.2.4 建立 TCP 通信与接收组帧

start_application() 创建 TCP PCB，绑定端口 5000，进入监听模式，再注册 accept_callback。连接建立后，accept_callback 为新连接注册接收及错误回调，并将其指针保存到 serpcb。

```
pcb = tcp_new_ip_type(IPADDR_TYPE_ANY);  
err = tcp_bind(pcb, IP_ANY_TYPE, 5000);  
pcb = tcp_listen(pcb);  
tcp_accept(pcb, accept_callback);
```

本工程使用单个全局连接。新的客户端连接会替换已有连接，因此调试助手和 Qt 上位机不要同时连接开发板。断开或发生连接错误时，程序清理连接指针与未收满的命令状态。

1.3.2.5 TCP 按字节流处理

TCP 回调边界不是命令边界。echo.c 遍历全部 pbuf 节点，将每个字节交给 command_stream_byte，保留未完成命令的位置，处理拆包和多条命令合并的情况。

```
tcp_recved(tpcb, p->tot_len);  
for (q = p; q != NULL; q = q->next) {  
    unsigned char *payload = (unsigned char *)q->payload;  
    for (i = 0; i < q->len; ++i)  
        command_stream_byte(payload[i]);  
}  
pbuf_free(p);
```

组帧器先寻找 0x55，再寻找 0xA5；帧头成立后收满 8 字节，交由 cmd_analysis() 检查帧尾 0xF0 并解析。未收满的字节保留到下一次回调。

表 1-24 组帧器函数说明

函数	作用
tcp_recv()	注册接收回调
tcp_recved()	通知协议栈已处理的接收字节数
tcp_sndbuf()	查询当前可用发送空间
pbuf_free(p)	释放本次收到的 pbuf

注意：采集或原始数据发送期间，cmd_analysis() 忽略新的控制命令，避免应答混入采样字节流。控制请求应在上一轮数据接收完成后发送。

1.3.2.6 TCP 控制命令格式

所有请求为 8 字节，帧头为 55 A5，帧尾为 F0。采集参数按 32 位大端数放在字节 3~6；寄存器读写命令则使用地址和数据字段。发送时选择 HEX 字节模式，不附加回车换行。

采集参数 : 55 A5 CMD P3 P2 P1 P0 F0
寄存器命令: 55 A5 CMD 00 AH AL DATA F0

表 1-25 TCP 命令表

CMD	功能	参数及返回
0x00	启动采集	按已设置参数启动；没有单独 8 字节启动 ACK
0x01	数据源选择	字节 6：0 测试、1 通道 1、2 通道 2；不使用 3
0x02	采集长度	字节 3~6 为总字节数，要求 16 字节对齐
0x03	速度接口参数	保存 Speed_Set，当前不进行数据抽取
0x04	兼容寄存器访问	参数为 reg_conf；不返回 0x81/0x84 应答
0x05	PLL 配置入口	交给 clk_wiz_Config；本章保持默认配置
0x10	写 ADC 寄存器	AH/AL 为地址，DATA 为写入值；返回 0x84
0x11	读 ADC 寄存器	AH/AL 为地址，DATA 请求值置 0；返回 0x81
0x12	查询训练结果	返回 done、pass 和最终 tap；不是扫描触发命令
0x1F	PING	返回 0x80，应答数据为 0x5A
0xFF	查询 ADC 设备信息	返回 48 字节信息结构体

33

寄存器请求的字节 3 置 0，地址由 AH、AL 组成。PS 实际送入 SPI 前将地址屏蔽为 13 位，所以有效地址范围不超过 0x1FFF。

设置长度时，N 个单通道样点对应 2N 字节。例如 4096 点需要发送 8192，即参数 00 00 20 00；不是直接发送点数 4096。通道与长度等采集参数命令没有逐条应答，配置后最后发送启动命令。

1.3.2.7 控制应答、训练状态与错误码

寄存器读写、tap 查询与 PING 使用统一的 8 字节应答。接收端应检查帧头、应答类型、状态、地址与帧尾，而不是只检查最后一个数据字节。

55 A5 RSP STATUS AH AL DATA F0

表 1-26 应答类型解释

RSP	含义	STATUS / DATA
0x80	PING 响应	STATUS=0, DATA=0x5A
0x81	寄存器读成功	STATUS=0, DATA 为读回值
0x84	寄存器写成功	STATUS=0, DATA 回显请求值
0x85	训练结果	STATUS bit0=done, bit1=pass; DATA=最终 tap
0xE0	命令错误	STATUS 为错误码

表 1-27 错误码解释

错误码	含义	处理方向
0x03	不支持的访问操作	检查寄存器访问操作码；未知顶层命令可能直接被忽略
0x04	约 2 秒等待超时	检查 DCO、SPI、硬件忙状态
0x05	硬件访问未就绪	等待上电和正常模式恢复完成
0x06	已有待处理寄存器事务	等待前一条应答，不并发下发
0x07	硬件命令失败	结合训练通过位和相关状态进一步定位

34

PING 请求: 55 A5 1F 00 00 00 00 F0
 PING 响应: 55 A5 80 00 00 00 5A F0
 读 0x0D : 55 A5 11 00 00 0D 00 F0
 正常模式: 55 A5 81 00 00 0D 00 F0
 查询 tap: 55 A5 12 00 00 00 00 F0
 结果示例: 55 A5 85 03 00 00 1D F0

结果示例表示 done=1、pass=1、tap=29。29 只是解码示例，不是固定结果。tap 响应的 STATUS=0x03 表示完成且通过，不能用普通寄存器应答“STATUS 必须等于 0”的规则判断它。

1.3.2.8 命令服务与设备信息

CM3432_CommandService() 在主循环中处理寄存器事务。它先尝试发送待发送应答，再检查命令完成翻转位；命令成功后，读操作从 CM3432_READBACK 取得值，写操作回显请求值，失败或超时则返回错误。

硬件完成事件由 DONE_TOGGLE 与提交前保存的 cm_done_before 比较。寄存器读写采用一条请求对应一个应答的方式；应答缓存只有一个槽位，上位机必须先收齐上一条应答，再进行下一条操作。

1.3.2.9 ADC 信息查询

0xFF 返回 ADC_Info.h 中定义的 packed 结构体，总长 48 字节，不带 55 A5 帧头。整数与 32 位浮点字段使用本工程 ARM 小端内存布局。

表 1-2848 字节设备信息布局

偏移	字段	值
0	ID, u32	1
4	Label, 16 字节	CM3432, 剩余字节补 0
20	Max_Sample, u32	250000000
24	Channel_Num, u8	2
25	Data_Type, u8	1, 低 14 位数据为补码
26	Resolution, u8	14
27	mode, u8	1
28	reserved, u32	0
32 / 36	Max_Volta / Min_Volta, float	5.0 / -5.0
40	Clk_Freq, u32	250000000
44	Cycle, float	4, 单位 ns

35

Channel_Num=2 表示模块有两路模拟输入，不表示当前 DMA/TCP 支持两路同时传输。设备信息中的时钟和量程为静态参数，运行时改 ADC 配置后不会自动刷新。

注意：PC 应区分三种接收阶段：8 字节控制应答、48 字节设备信息、按已知长度接收的采样数据。不要在原始采样数据中搜索 55 A5 作为帧边界。

1.3.2.10 采样启动流程

主循环持续处理网络输入和定时事件，同时调用设备信息、寄存器服务、采样启动和数据发送函数。Restart_ADC_Sample 的参数就是命令 0x02 设置的总字节数。

```
xemacif_input(echo_netif);  
Send_ADC_Info();  
clk_wiz_Config(PLL_Reconfig);  
CM3432_CommandService();  
Restart_ADC_Sample(DataNum);  
Data_Recive_Trans();
```

启动函数检查 restart、采集忙状态、DATA_READY、TRAIN_PASS、待处理寄存器事务、TCP 连接，以及长度非零、16 字节对齐和软件容量上限。训练尚未满足时先等待，不立即开放 PL 数据。

1.3.2.11 先准备 DMA，再使能采样

```
ADC_SAMPLE_CTRL_0_mWriteReg(SAMPLE_CTRL_ADDR, DATA_SUM, Bytes);
ADC_SAMPLE_CTRL_0_mWriteReg(SAMPLE_CTRL_ADDR, CH_SEL, ChannelSel);
ADC_SAMPLE_CTRL_0_mWriteReg(SAMPLE_CTRL_ADDR, SPEED_CTRL, Speed_Set);
ADC_SAMPLE_CTRL_0_mWriteReg(SAMPLE_CTRL_ADDR, BYTES_Trans, chunk);
ADC_SAMPLE_CTRL_0_mWriteReg(SAMPLE_CTRL_ADDR, SAMPLE_EN_GO, 0x100);

if (start_trans_data(chunk, RX_BUFFER_BASE) != XST_SUCCESS) {
    abort_capture();
    return;
}
set_dma_timeout(chunk);
ADC_SAMPLE_CTRL_0_mWriteReg(SAMPLE_CTRL_ADDR, SAMPLE_EN_GO, 0x101);
```

chunk 取总字节数与 0x03FF0000 的较小值。0x100 只关闭采样位，Go 仍为 1；提交 S2MM 地址和长度成功后，再写 0x101 开放采样。采集过程中不重新初始化 ADC，也不重新执行上电扫 tap。

注意：0x00 启动命令没有独立的 8 字节成功应答。发出启动后，PC 应进入原始数据接收阶段并设置超时，不应等待 0x84。长度错误、未连接等拒绝原因会通过串口诊断输出。

1.3.2.12 DMA 中断与分段接收

start_trans_data() 禁用 DCache，清理 DMA 接收标志，提交 S2MM 传输，再使能完成与错误中断。DMA 将采样数据写到指定 DDR 地址，不由 CPU 逐点搬运。

```
Status = XAxiDma_SimpleTransfer(&AxiDmaInstance,
    RxBufferPtr, len, XAXIDMA_DEVICE_TO_DMA);
if (Status != XST_SUCCESS)
    return XST_FAILURE;
XAxiDma_IntrEnable(&AxiDmaInstance,
    XAXIDMA_IRQ_IOC_MASK | XAXIDMA_IRQ_ERROR_MASK,
    XAXIDMA_DEVICE_TO_DMA);
```

1.3.2.13 中断处理只记录状态

platform_zynq.c 中的 RxIntrHandler 读取并确认 S2MM 中断。错误时置 Error，完成时置 RxDone，具体的后续 DMA 和 TCP 工作放在主循环处理。

```
IrqStatus = XAxiDma_IntrGetIrq(AxiDmaInst,
    XAXIDMA_DEVICE_TO_DMA);
XAxiDma_IntrAckIrq(AxiDmaInst, IrqStatus,
    XAXIDMA_DEVICE_TO_DMA);
if (IrqStatus & XAXIDMA_IRQ_ERROR_MASK) {
    Error = 1;
    return;
}
```

```
}  
if (IrqStatus & XAXIDMA_IRQ_IOC_MASK)  
    RxDone = 1;
```

1.3.2.14主循环衔接下一段

Data_Recv_Trans() 检测到 RxDone 后，关闭该段中断并对已写入 DDR 范围执行缓存失效。若 data_rest 不为 0，则 RX_BUFFER_BASE 增加实际完成段长度，重新计算下一段大小，提交 DMA 后再次开放采样。

全部段接收完成后，SAMPLE_EN_GO 清零，恢复 DCache，将 capture_ready 置 1，进入 TCP 发送。发生 DMA 错误、连接丢失或等待超时，则 abort_capture() 关闭采样、复位 DMA、清理计数并中止连接。

1.3.2.15TCP 数据发送

本轮采样全部写入 DDR 后，PS 才通过 TCP 发送。每次主循环最多推进一个发送片段，长度取“剩余字节数、可用发送空间和 1460 字节”的最小值，以便持续处理网络输入和定时事件。

```
available = tcp_sndbuf(serpcb);  
if (available > 0 && send_data > 0) {  
    chunk = (send_data > TCP_CHUNK_BYTES)  
        ? TCP_CHUNK_BYTES : send_data;  
    if (chunk > available)  
        chunk = available;  
    err = tcp_write(serpcb,  
        (void *)(ADC_Data_Addr + uiIdx), chunk,  
        TCP_WRITE_FLAG_COPY |  
        ((chunk < send_data) ? TCP_WRITE_FLAG_MORE : 0));  
    if (err == ERR_OK) {  
        send_data -= chunk;  
        uiIdx += chunk;  
        tcp_output(serpcb);  
        XTime_GetTime(&send_progress_time);  
    } else if (err != ERR_MEM) {  
        abort_capture();  
        return;  
    }  
}
```

只有 tcp_write 返回 ERR_OK 才推进发送位置；ERR_MEM 时保留当前位置，等待后续主循环重试，其他错误则中止。约 30 秒没有成功入队进展时，程序执行发送超时处理。

send_data 减为 0 表示所有字节已交给 lwIP 发送队列，PC 仍需累计接收至本轮约定总字节数。程序没有“PC 已保存文件”的应用层确认，因此文件长度检查由上位机完成。

注意：一次 tcp_write 不对应 PC 的一次 readyRead 或 recv。1460 字节也不是样点帧头长度或 DMA 段长度。接收端只按累积字节数组织本轮数据，不按网络回调次数判断采集是否完整。

1.4 板级验证

1.4.1 实验所需硬件

1. [AC820 开发板](#) x1
2. [ACFL3432 模块](#) x1
3. 信号发生器 x1
4. 千兆网线 x1
5. DC 电源线 x1
6. Type-c 下载线 x1

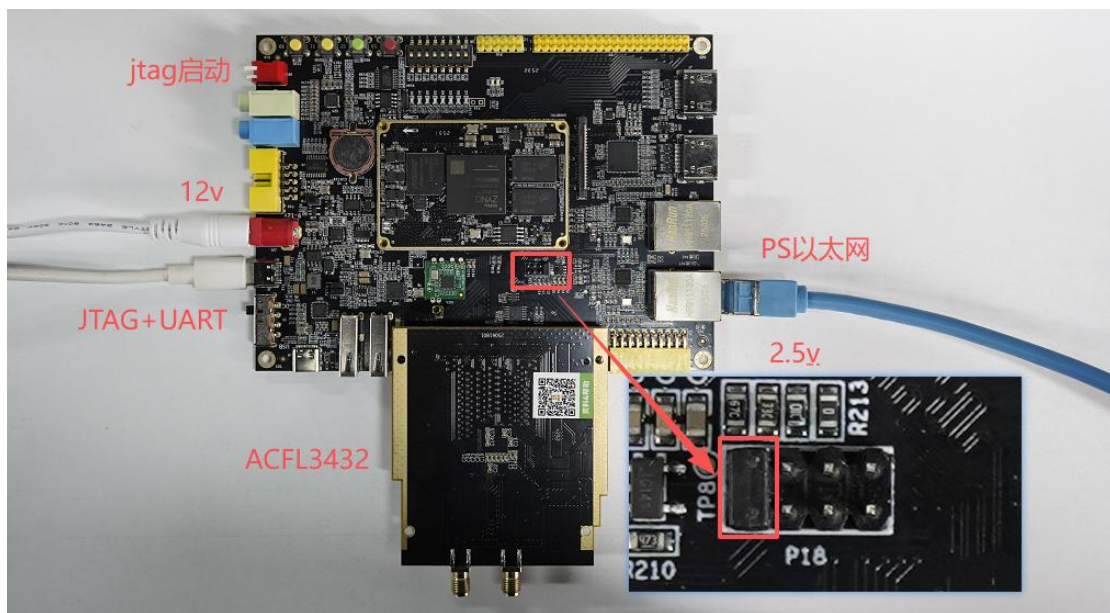


图 1-20 硬件连接图

在断电状态下，将 ACFL3432 插入开发板 FMC_LPC 接口，确认插接方向和固定位置。根据工程要求将相关供电跳线设为 2.5 V，并核对主板与采集模块的供电连接。不要在上电状态下插拔 FMC 模块或切换供电跳线。

千兆网线一端连接开发板 PS 侧网口（开发板背面丝印 PS_ETH），另一端连接 PC；下载器和 PS 串口连接电脑。将信号源接入待测试的模拟通道，明确记录波形类型、频率、峰峰值 V_{pp} 和直流偏置，然后再给开发板上电。

1.4.2 修改电脑 IP 地址

在 LWIP 功能初始化小节中，我们通过代码将开发板 IP 地址设置为了 192.168.0.2，因此，我们需要将电脑 IP 地址设为同一网段。

在电脑上进入【控制面板】->【网络和 Internet】->【查看网络状态和任务】，查看网络连接状态。需要看到在活动网络中有以太网连接存在，才表明开发板和电脑的网络才已经连通。至于显示的无法连接到网络选项，意思是指无法连接到互联网获取网络上的数据，这是正常的，无需在意，如图：



图 1-21 查看网络连接状态

点击“以太网”文字，以查看该网络状态，确认当前连接速度为千兆速率（1000.0 Mbps），如图：



图 1-22 查看网络速度

在上述本地连接状态中，点击属性，并在弹出的属性对话框中双击【Internet 协议版本 4（TCP/IPv4）】选项，然后在弹出的属性对话框中设置静态 IP 地址（默认网关可以不设置）。如图：

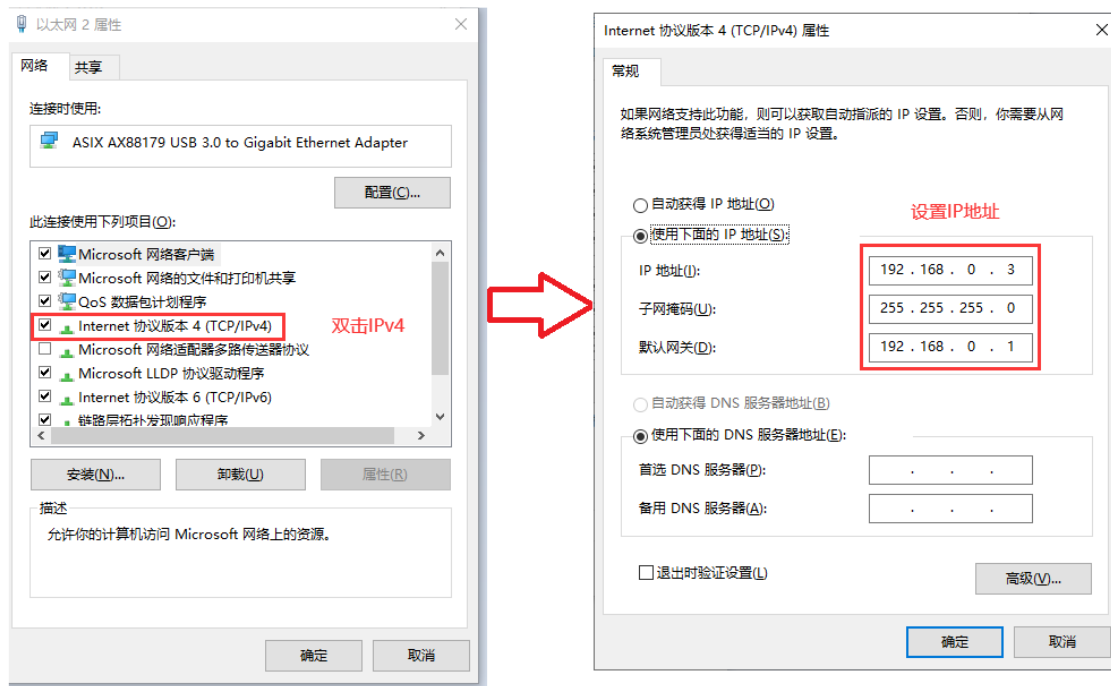


图 1-23 修改 PC 的 IP 地址

如果用户电脑中相关驱动缺失文件，会出现弹窗报错“出现了一个意外的情况，不能完成所有你在设置中所要求的更改。”导致修改失败，对于该情况用户可以参考下帖：

[设置静态 IP 时提示“出现了一个意外的情况，不能完成所有你在设置中所要求的更改”](#)

更改完 IP 地址后，还需要检查是否有 arp 绑定，以管理员权限打开 CMD，输入以下命令查询 arp：

```
arp -a
```

查询界面如图：

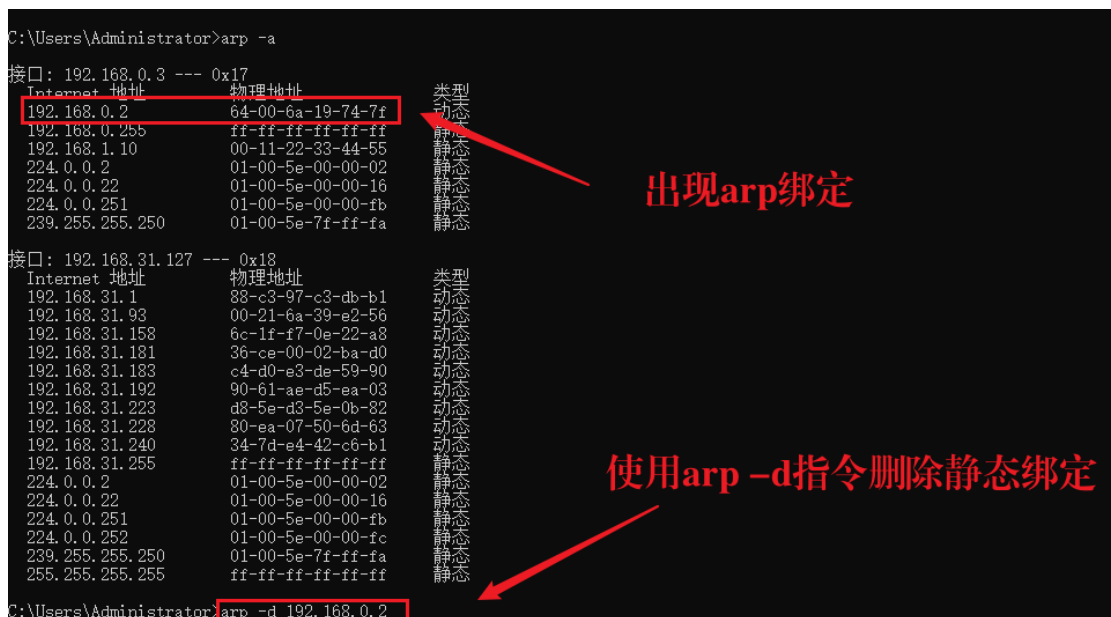


图 1-24arp 查询与删除

如果出现了图中的静态绑定，则需要使用以下指令进行删除：

```
arp -d 192.168.0.2
```

1.4.3 代码烧录

由于 LWIP 的初始化需要一段时间，因此在烧录代码之前，我们需要在 SDK 终端中连接串口，通过串口打印的信息观察。串口的端口号需要通过设备管理查询，在端口栏中，名字为 CH9102 的便是 PS 端串口。譬如，这里笔者电脑上分配给开发板 PS 端串口的端口号为 COM60。

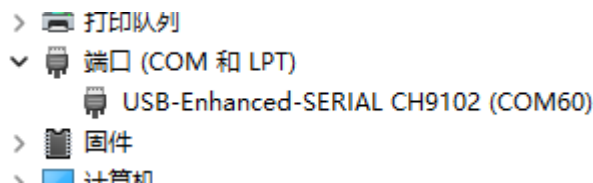


图 1-25 查询端口号

大家连接串口时，请以自己电脑上分配的端口号为准。连接完串口后，便可以烧录程序，LWIP 的初始化需要等待一段时间，当串口打印出如下图所示信息时，便代表初始化完成，此时便可以连接以太网上位机进行功能验证。

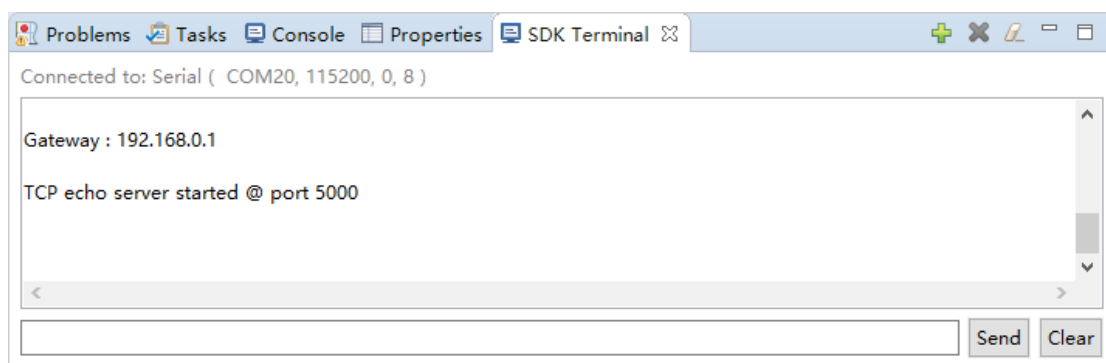


图 1-26 LWIP 初始化完成

或者使用串口调试助手进行信息打印，最后出现“TCP echo server started @ port 5000”即为 TCP 协商通过。

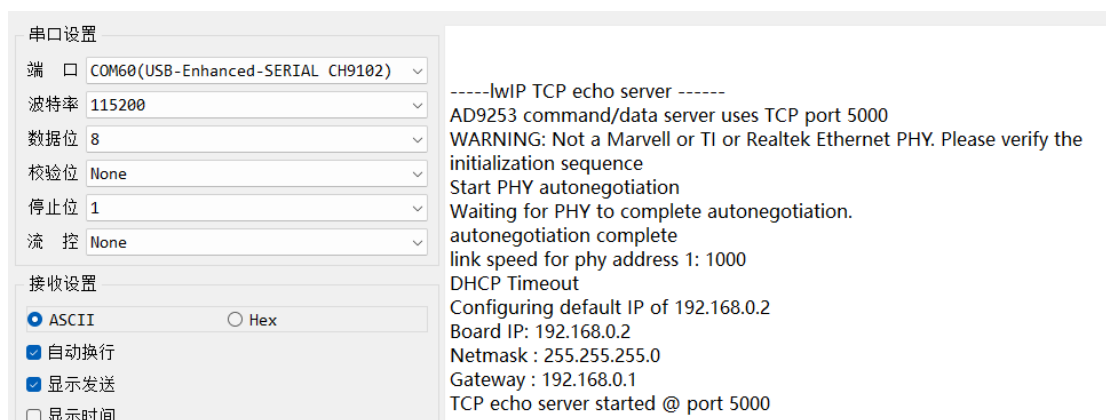


图 1-27 串口调试助手信息打印

1.4.4 功能验证

1.4.4.1 寄存器访问与自动训练

以下操作在采集空闲状态下执行。每发送一条有应答的命令，先收齐 8 字节，检查类型 and 状态，再发送下一条。

表 1-29 寄存器与训练检查

操作	发送字节	预期检查内容
PING	55 A5 1F 00 00 00 00 F0	RSP=0x80, DATA=0x5A
查询训练	55 A5 12 00 00 00 00 F0	RSP=0x85, done=1、pass=1
读取测试模式	55 A5 11 00 00 0D 00 F0	正常采样时应读回 0x00
读取时钟配置	55 A5 11 00 00 0B 00 F0	默认初始化值为 0x00
重复读取	再次发送同一条读取命令	收到新的应答，确认重复请求可执行

这里暂时不使用小梅哥上位机，这里先使用网络调试助手进行指令发送和回传。

协议类型选择：TCP_Client；

远程主机地址：192.168.0.2；

远程主机端口：5000；

发送和接收选择：HEX。



图 1-28 网络调试助手设置



图 1-29 指令下发和回传

1.4.4.2 验证时钟配置变化后的重训练

先读取 0x000B，记录当前值和 tap。选用已确认合法的新时钟配置后，通过 0x10 写入，并等待写成功应答。若有效低 6 位发生变化，控制器将进入棋盘格、应用新值、重新扫描，再退出测试模式。

写 0x0B 的格式：55 A5 10 00 00 0B DD F0
恢复默认配置：55 A5 10 00 00 0B 00 F0
随后查询训练：55 A5 12 00 00 00 00 F0
检查正常模式：55 A5 11 00 00 0D 00 F0

DD 表示按芯片寄存器定义选择的数据字节，不是固定字符。应答成功后，确认 TRAIN_DONE、TRAIN_PASS 和 0x000D 读回值，再开始采样。恢复默认值时同样等待整个事务完成。

重复写入已经成功训练的同一低 6 位配置时，不再重新扫描。需要观察路径差别时，在硬件调试中查看 runtime_retrain_busy、retrain_toggle、training_done 和 selected_tap；最终 tap 相同不能说明一定没有执行扫描。

注意：0x12 只读取训练结果，没有强制扫描或手动设 tap 功能。出现超时或失败应答时先停止采样操作，记录完整应答和串口信息，再检查相关时钟与状态。



图 1-301.4.4.2验证时钟配置变化后的重训练

1.4.4.3单通道采集命令实例

以通道 1 采集 4096 点为例，每点 2 字节，总长度为 8192 字节，即 0x00002000。先完成 PING、训练查询及正常模式检查，并确保没有尚未读取的控制应答。

选择通道 1: 55 A5 01 00 00 00 01 F0
设置字节数: 55 A5 02 00 00 20 00 F0
速度字段清零: 55 A5 03 00 00 00 00 F0
启动采集: 55 A5 00 00 00 00 00 F0

以上参数和启动命令不返回逐条成功应答。启动后按原始采样数据累计接收 8192 字节，得到 4096 个样点。PC 不能把第一次收到的数据长度当作整轮长度，也不应继续向同一连接插入 PING 或寄存器查询。



图 1-31 单通道采集命令实例

1.4.4.4 验证测试数据

将通道选择命令的字节 6 改为 0，可采集 FPGA 递增测试计数。以无符号方式取低 14 位，检查相邻值按模 16384 递增。首个样点不强制等于 0，重点检查同一 128 位数据拍内的 8 个样点和相邻拍之间的顺序。

```
raw14 = word16 & 0x3FFF  
next_expected = (raw14 + 1) & 0x3FFF
```

测试计数用于验证通道选择之后的 FIFO、DMA、TCP 与 PC 解码链路，不能替代模拟输入质量验证。该数据源仍经过本工程采集门控，需要训练通过后才能启动。

注意：单次采集结束后，再把通道改为 2 启动另一轮。两轮分开采集不能用于判断同一时刻两路信号的相位关系。

1.4.4.5 Qt 上位机连接与显示设置

打开“小梅哥数据采集仪”，进入通信设置，选择以太网 TCP。填写串口打印的开发板 IP，端口设为 5000；本机地址选择实际连接开发板的网卡。下面是通信设置界面示例，界面布局随软件版本而不同。

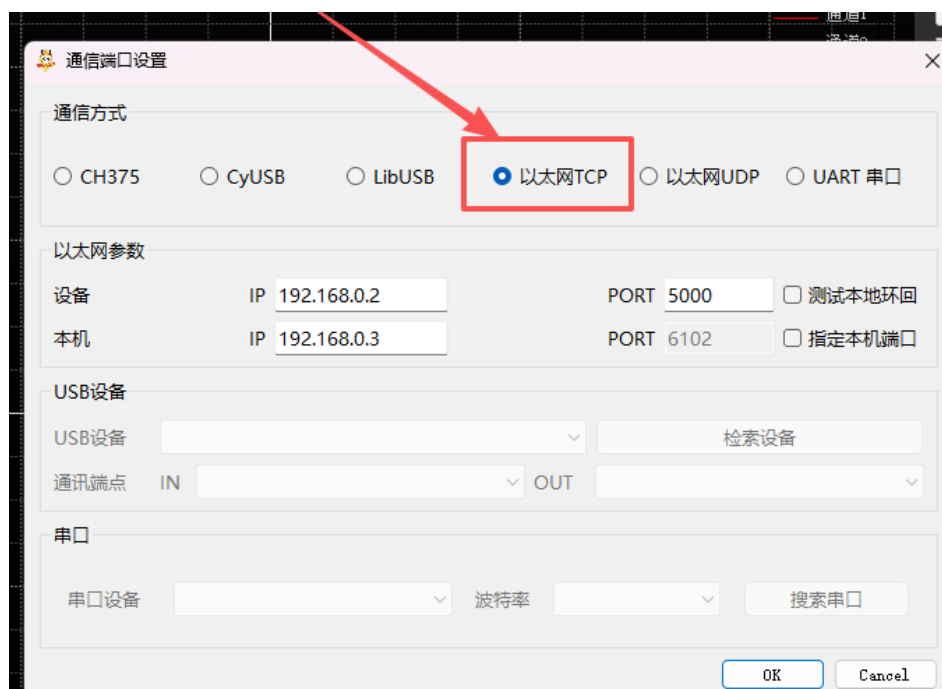


图 1-32 上位机以太网 TCP 通信设置

设备型号选择 CM3432，分辨率为 14 位，输入范围按模块标定设置。

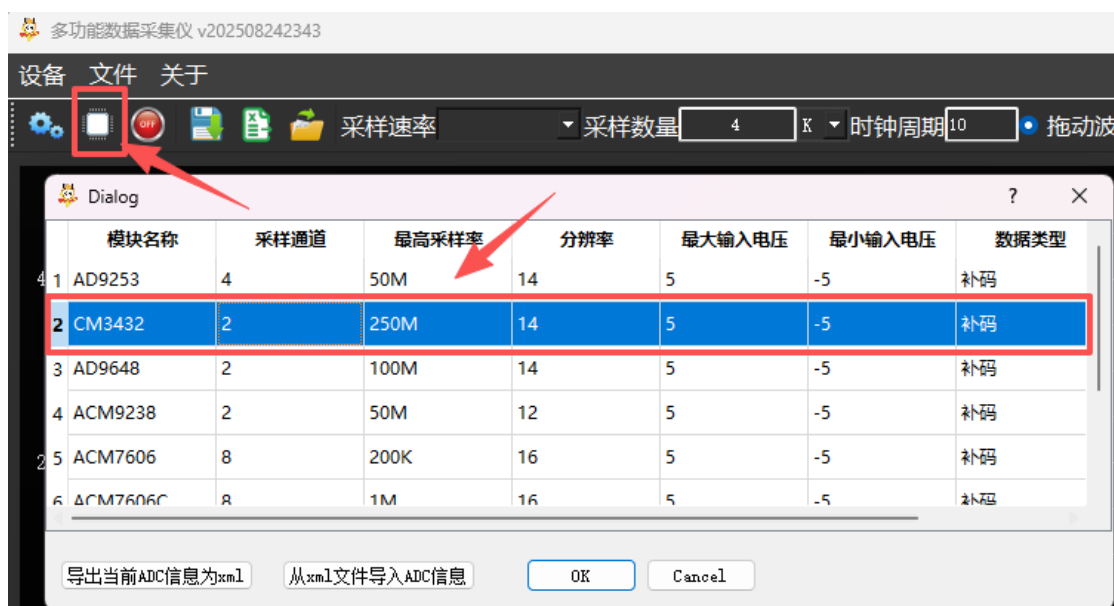


图 1-33 选择 ADC 型号

采集源一次只选择通道 1、通道 2 或测试计数；点数在发送前转换为字节数，例如 4K 点发送 8192 字节。

表 1-30 小梅哥上位机设置

界面项目	设置要求
通信方式	以太网 TCP，实际 Board IP，端口 5000

界面项目	设置要求
通道	单选通道 1 / 通道 2 / 测试源
采集长度	N 点转换为 2N 字节发送 0x02
采样率	默认 250 MSPS；改变 ADC 时钟后同步更新横轴
数据格式	小端 16 位容器，低 14 位补码，高两位补 0
控制操作	使用 0x10/0x11/0x12，完整处理 8 字节应答

1.4.4.6 模拟信号采集与波形检查

接入已知模拟信号。先只选择一个通道进行采集，再切换另一通道重复操作。测试记录应包含波形类型、频率、V_{pp}、直流偏置、采样时钟配置和训练结果。

表 1-31 模拟信号验证步骤

测试项	操作示例	检查内容
通道 1	接入 100 kHz 三角波，明确设置 V _{pp} 和偏置	波形类型、周期、正负半周和削顶情况
通道 2	接入 200 kHz 正弦波，保持输入在允许范围	正弦形状、周期和电压幅度
点数	每轮设置 4096 点，即 8192 字节	接收字节数与解码点数
训练状态	采样前查询 done/pass/tap	done=1、pass=1，正常模式已恢复
幅度	记录信号源输出负载设置与实测输入	V _{pp} 与上位机换算结果相符

47

在默认 250 MSPS 下，100 kHz 周期为 10 μs，约占 2500 个样点；200 kHz 周期为 5 μs，约占 1250 个样点。这些是参数计算关系，用于检查时间轴，不是预先给定的测量结果。

1.4.4.7 实际测量

可以看到，通道 1 采集到的是三角波对应的峰峰值为 4.9256V 属于正常范围内。同时，默认设置的采样数量为 4K，每个数据大小为 2 字节，因此单通道情况下一共需要采集 4*1024*2*1=8192 字节，实际获取到的字节数与该数量一致。

经过测量，A1 和 A2 之间间隔 10000ns，也就是 100Khz。

综上所述，采集绘制的通道 1 为 100Khz 的三角波，V_{pp} 为 4.9146V；并且波形完整，并未出现毛刺现象。

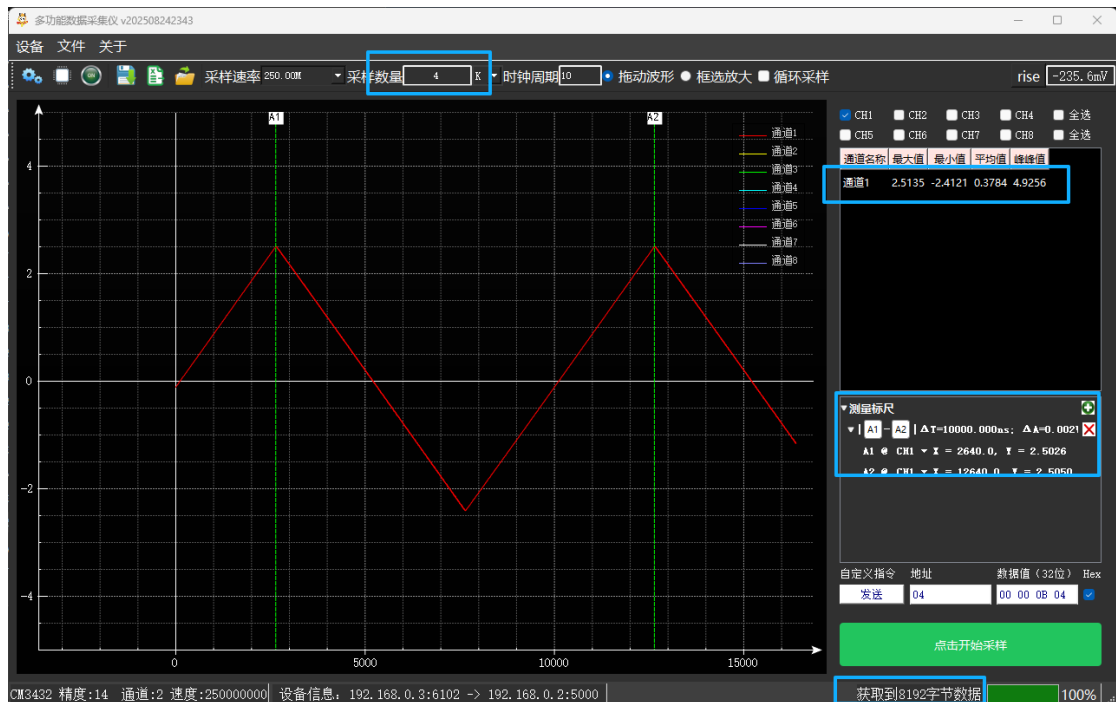


图 1-34 第一次采集波形

再来测试一下大数据量时的表现（数据字节数大于一次 DMA 中断的最大值，即大于 64MB），这里设置采样数量为 40M，双通道的情况下对应需要采集 80MB 字节数据。

选择通道 2，采集数量修改为 40M 进行采样，数量很大，波形会挤在一起显示，最后上位机采集数量为 83886080。计算一共需要采集 $40 \times 1024 \times 1024 \times 2 \times 1 = 83886080$ 字节，实际获取到的字节数与该数量一致。

48

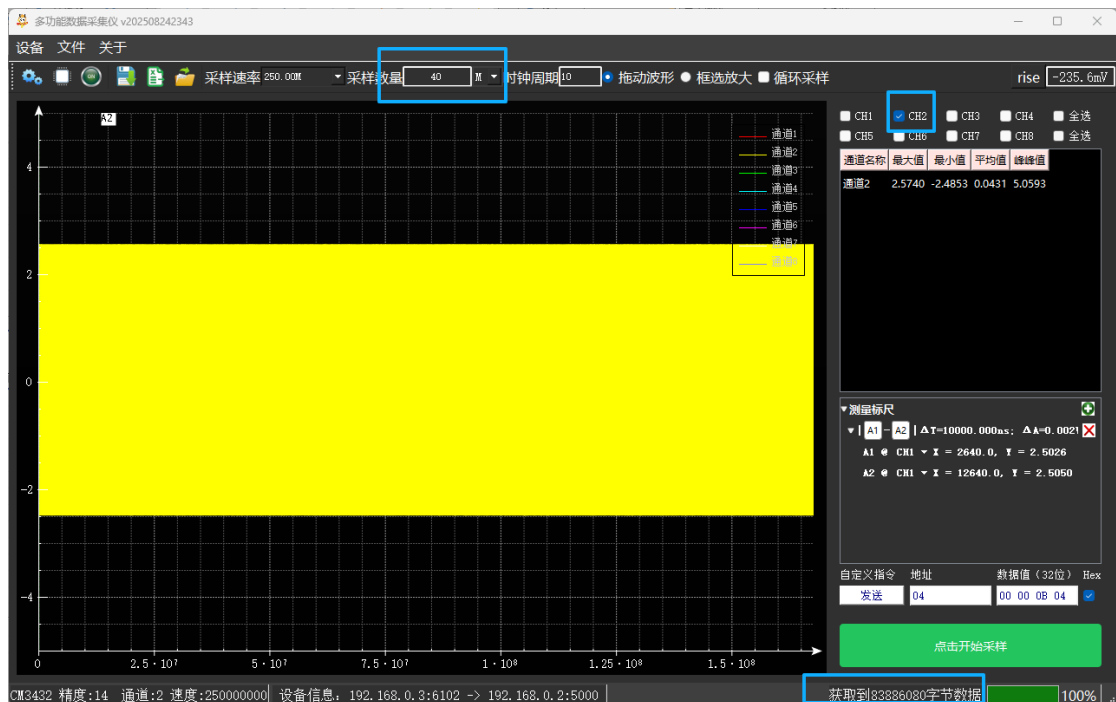


图 1-35 第二次采集波形

放大进行测量，测量结果为 5000ns，也就是 200KHz。峰峰值为 5.0593V。

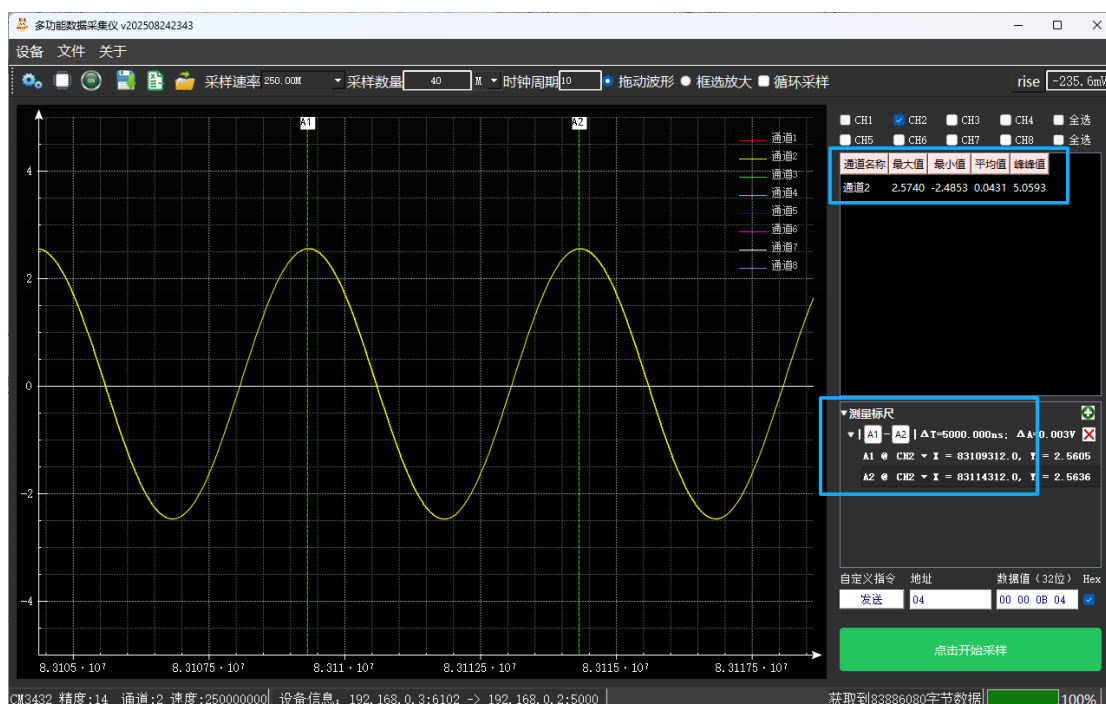


图 1-36 第二次采集波形放大测量

然后可以测试上位机发送修改分频指令进行采集测量。例如写入 0x000B 的分频配置后，PL 会自动执行重训练；上位机的采样率也必须按 250 MHz 基准和实际分频比同步修改，再开始采样。频率测量应以波形周期是否与信号源一致为判断依据。

比如这里输入地址 0x04，数据值“00 00 0B 04”，也就是进行 5 分频，点击发送，对于 250mhz 采样率 5 分频为 50mhz，那么上面的采样速率也需要同步修改为 50mhz，最后点击开始采样。测量结果依旧是 5000ns，也就是 200Khz。

49

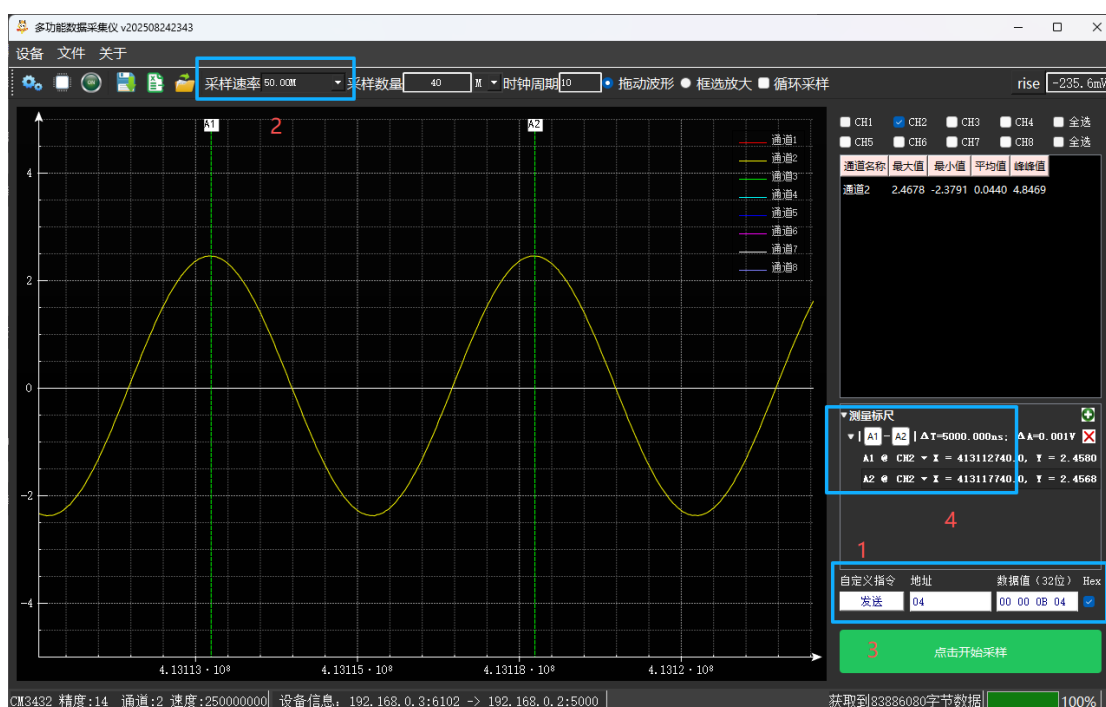


图 1-37 5 分频采集测量

综上，也就说明设计无误，代码能够按照预期运行。

1.5 思考与总结

本章将 CM3432 寄存器配置、公共 tap 自动训练、单通道采样、DMA 缓存与 TCP 传输连接成完整系统。阅读代码时，应沿着“时钟与供电—SPI—接收训练—采样门控—FIFO/DMA—TCP—PC 解码”的顺序理解各模块之间的关系。

表 1-32 常见问题定位

现象	优先检查
TCP 无法连接	串口实际 IP、PS 网口、端口 5000、DHCP、PHY 与网卡配置
PING 正常但 done=0	返回 DCO、200 MHz 参考、IDELAYCTRL RDY、PDWN 和启动等待
done=1、pass=0	棋盘格配置、LVDS 连接、接收时钟与可用 tap 窗口
写 0x000B 后超时	SPI 完成事件、DCO 是否运行、重训练等待状态
启动采集后没有数据	单通道选择、总字节数、16 字节对齐、训练门控与短帧条件
负半周或零点附近异常	是否按低 14 位补码解释，而非直接转为有符号 16 位
修改界面速率但波形不变	0x03 不进行抽取；核实际 ADC 时钟和显示横轴
小数据正常，大数据异常	DDR 区域、DMA 分段、FIFO 满、TCP 发送超时
采样开头错位	是否存在未读完的 8 字节应答或 48 字节设备信息
连接另一客户端后异常	当前是单客户端连接，不支持独立并行会话

50

操作时记住三个条件：

- 第一，采样前应完成训练并恢复正常模式，TRAIN_PASS 不能省略；
- 第二，采样命令传递的是总字节数，单路 N 点对应 2N 字节；
- 第三，上位机必须按小端 16 位容器中的 14 位补码恢复数据，并使用真实采样率建立横轴。

自动扫 tap、硬件数据缓存和 TCP 可靠传送各解决一类问题，不能互相替代。调试时保存命令、状态和原始数据，逐层验证，再判断问题发生在哪一段链路。

最后，这里再为大家补充两个可能遇到的问题：

1. 无法修改 IP 为静态 IP 地址

出现这种情况，通常都是相关驱动缺少部分文件造成的，此时我们可以使用管理员权限打开 cmd，在其中输入以下代码来修改静态 IP：

```
netsh interface ip set address "以太网名" static 静态 IP 255.255.255.0
```

其中，红色部分为需要用户自己输入的部分，以太网名通常由“以太网”+“空格”+“数字”组成，用户输入时，注意不要忘记空格。

如果想将 IP 恢复到动态 IP 地址，只需要使用以下语句：

```
netsh interface ip set address name="以太网名" source=dhcp
```

2. 修改LwIP源工程模板（针对rt18211f以太网芯片）

前面我们对模板例程的修改，只是针对本次创建的模板例程而言的。一旦我们重新生成 bsp，或者新建一个同样模板工程时，都需要再次修改。

对此，我们可以直接修改工程模板的源文件（修改前请一定要进行备份，方便恢复），具体步骤如下：

- a) 找到 Vivado 安装目录下的 Xilinx\SDK\2018.3\data\embeddedsw\ThirdParty\sw_services\lwip202_v1_2\src\contrib\ports\xilinx\netif 路径下的 xemacpsif_physpeed.c 文件
- b) 按照前文中的修改方法来修改它，然后保存。修改完成后，新创建的 LwIP 模板例程，都不需要再进行修改。