

AMKN 系列工控板 软件测试说明

(2026 年 7 月 24 日修订版)

版权声明

本产品使用手册包含的所有内容均受版权法的保护，未经北京中嵌凌云电子有限公司的书面授权，任何组织和个人不得以任何形式或手段对整个手册和部分内容进行复制和转载。

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。除在其产品的销售条款和条件声明的责任之外，我司概不承担其他责任。并且我司对本产品的销售和使用不作任何明示或暗示的担保，包括对产品特定用途的适用性，适销性或对任何专利权、版权或其他知识产权的侵权责任等均不作担保。我司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，我司可能随时会对产品描述和相关的功能调整或技术改进，保留修改文档中任何内容的权利，恕不另行通知。

商标声明

AMKN 系北京中嵌凌云电子有限公司注册商标，未经书面授权，任何人不得以任何方式使用该商标、标记。

销售及服务网络

北京

销售电话：188 0108 0298

地址：北京市海淀区吴家场路 1 号院 2 号楼

邮箱：SALES@EMBEDARM.COM

西安

销售电话：189 9285 2102

地址：西安市曲江新区旺座曲江 H 座 3003 室

邮箱：SALES@EMBEDARM.COM

技术支持：

电话：029-8877 2044（工作日）

手机/微信：133 9928 8868

邮箱：EMBEDARM@126.COM

网址：WWW.ZQLYMCU.COM

版本

表格显示本产品使用手册在不同时期的修订版本及修订原因说明：

版本	修改内容	完成日期	修订部门
V1.00	正式发布	2026. 7. 24	研发部

适用产品型号：

序号	类型	订货型号	备注
1	工业控制模块	STM32F103VE、STM32F103ZE、STM32F107VC、STM32F407VE、STM32F407ZE、GD32F303VE、GD32F303ZE、GD32F307VE、STM32H723ZG、STM32H743XI、STM32MP157	
2	工业控制板	AMKN8602、AMKN8602G、AMKN8626、AMKN8628、AMKN8629、AMKN8630、AMKN8632、AMKN8638、AMKN8639、AMKN8701G、AMKN8702G、AMKN8703、AMKN8804、RTU-BUS、RTU-6XXX系列	

特别提醒：本软件只适用于以上产品硬件，并不适用于其它产品。

目录

第一章. AMKN 软件框架概述	5
1. 开发框架选择	5
2. 驱动库对比	5
3. 软件框架对比	6
第二章. AMKN 软件框架例程基本操作	8
1. 打开软件	8
2. 插入 ST-LINK 仿真器并给板子加电, 如下图:	8
3. 配置 ST-LINK 仿真器, 如下图:	9
4. 编译软件并下载程序如下图:	10
5. 点击全速运行, 观察板子运行情况	11
第三章. AMKN 软件框架例程测试功能说明	12
1. RUN LED 运行状态指示	12
2. ALARM 蜂鸣器控制	12
3. 调试信息输出	12
4. DI 端口测试	14
5. DO 端口测试	14
6. PWM 输出测试	15
7. FCLK 频率测量	16
8. AI (ADC) 端口输入测试	17
9. AO (DAC) 端口输出测试	18
10. RS232 和 RS485 通信端口测试	19
11. CAN 通信端口测试	20
12. 网络通信测试	22
13. WIFI 通信测试	24
14. EEPROM 读写测试	26
15. SPI FLASH (25Q64) 读写测试	27
16. U 盘读写测试	28
17. SD 卡读写测试	29
18. SDRAM 读写测试	30
19. LCD 显示测试	31
附录: 文档修改记录	32

第一章. AMKN 软件框架概述

1. 开发框架选择

用户在购买我们的工控板后有两种开发平台可以选择：中嵌凌云的 AMKN 软件框架和 ST 原厂 HAL 库软件框架。该如何选择，简单理解如果用户非常熟悉 HAL 库软件开发就优先选择 HAL 库开发；如果不熟悉 HAL 库，可以选择中嵌凌云的 AMKN 软件框架。

2. 驱动库对比

AMKN 软件框架使用的是 AMKN 驱动库，ST 原厂 HAL 库软件框架使用是 HAL 驱动库。

下面先对比 AMKN 驱动库和 HAL 驱动库：

序号	对比项目	AMKN驱动库	HAL驱动库
1	是否开源	不开源，以xxx.lib和引用头文件形式提供	开源，以源代码形式提供
2	支持芯片	只能用以下芯片： STM32F103XX、STM32F107VC、 STM32F407XX、STM32H723XX、 STM32H743XX、STM32H750XX、 STM32H757XX、STM32MP15XXX、 STM32MP25XXX GD32F303XX、GD32F307XX、 GD32F407XX、GD32F470XX 持续增加中...	ST公司所有STM32芯片
3	接口函数特点	接口函数简洁，一般每个功能只提供4个接口函数，以UART为例： 初始化：Uart_Init(...) 读取：Uart_Read(...) 写入：Uart_Write(...) 控制：Uart_Ctrl(...)	接口函数众多，功能齐全
4	可移植性	高，同一套API接口适用所有支持芯片包括GD32系列芯片	高，同一套API接口可无缝迁移至F1/F4/F7/H7等不同系列芯片
5	统一抽象接口	屏蔽底层寄存器差异，不区分轮询、中断、DMA三种工作模式，由初始化配置，在函数内自动实现	屏蔽底层寄存器差异，提供轮询、中断、DMA三种工作模式，适配不同实时性与效率需求
6	效率	封装层级少，效率高	封装层级多，效率适中
7	对使用者要求	适中，函数比较少，接口简单	高，函数非常多，接口复杂

8	功能覆盖	大部分重要功能，可以覆盖HAL库95%的功能	全部功能
9	适用软件框架	AMKN软件框架	ST的HAL库软件框架
10	驱动库版本	V1.10/V1.20/V1.31/V1.40 最新版本:V1.40.00	STM32F1XX: V1.8.0 STM32F4XX: V1.25.0 STM32H7XX: V1.25.0
11	支持产品	只支持中嵌凌云AMKNxxxx系列工控板及工控模块	支持中嵌凌云AMKNxxxx系列工控板、工控模块和第三方的各种STM32工控板

3. 软件框架对比

项目	项目	AMKN软件框架(V1.40)	HAL库软件框架(V1.25)
1	适用驱动库	AMKN驱动库	HAL驱动库
2	是否开源	开源	开源
3	框架结构	IPO 数据流程控制包括： I 是 Input：数据输入和状态输入；P 是 Process：数据处理，回调用户应用程序； O 是 Output：数据输出和控制	无，需自己实现
4	操作系统	UCOS-II V2.86 FreeRTOS V10.5.1	FreeRTOS V10.3.1
5	文件系统	FatFS R0.13b	FatFS R0.12c
6	TCP/IP协议栈	LWIP-2.1.3	LWIP-2.1.2
7	OpenAMP	支持	支持
8	LVGL	支持	无
9	TFTP固件更新	支持	无
10	ModbusRTU ModbusTCP	支持	无
11	CANOpen	支持	无
12	步进电机控制 函数	支持	无
13	各种外围器件 驱动	支持，包括但不限于：TM1640、WTN6XXX、 DAC8562、MCP4728、DAC128S085、BY8302、	无

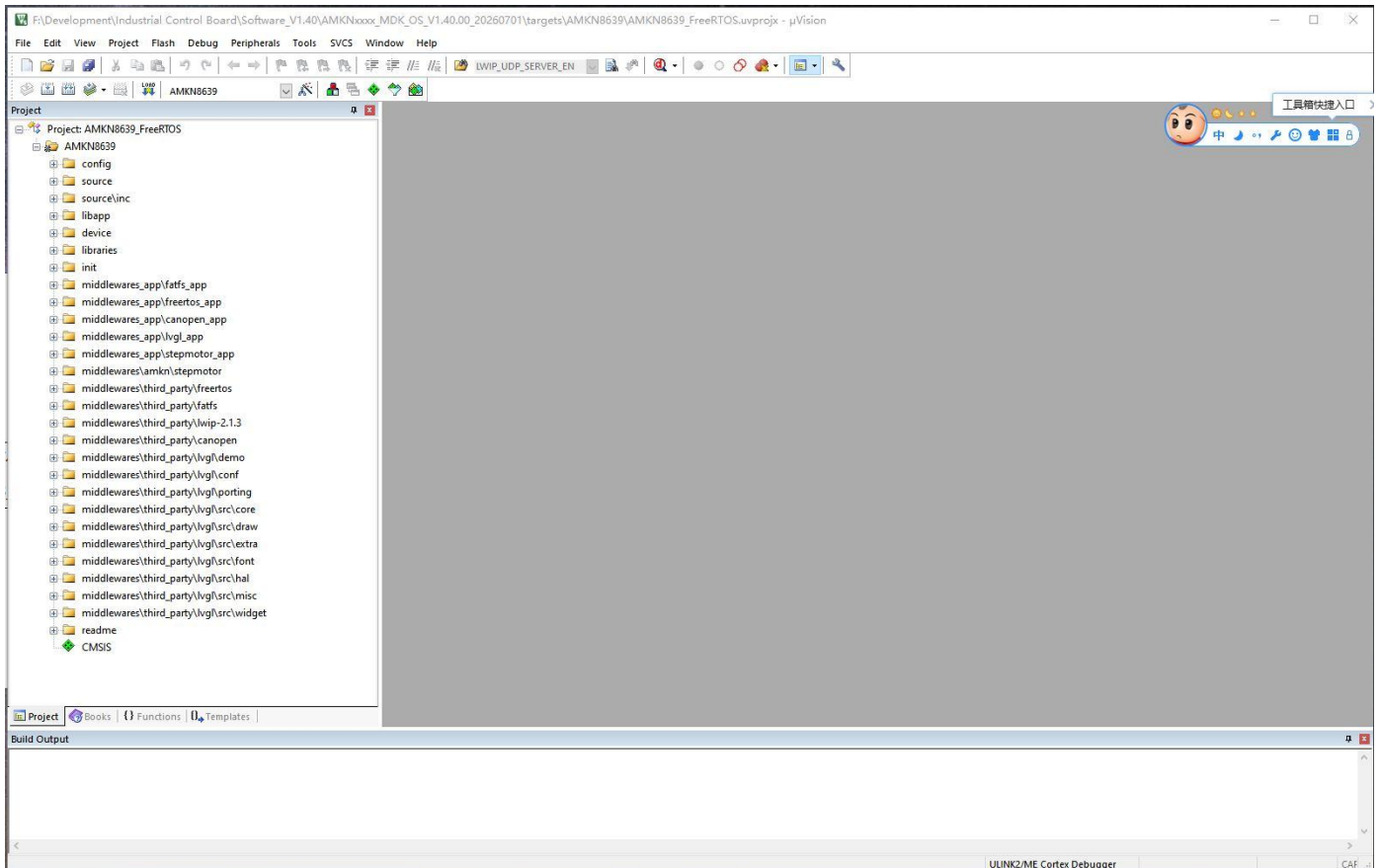
		WIFI模块、W5500、DAC8554、CH455、 ADS1220、ADS1232、AD7682、MCP3208、 AD5175、SHTXX、CH9434、MAX6675	
14	支持产品	只支持中嵌凌云AMKNxxxx系列工控板及 工控模块	支持中嵌凌云AMKNxxxx系列工控板、工 控模块和第三方的各种STM32工控板
15	开发环境	目前只支持Keil (MDK V5.XX)	支持Keil (MDK V5.XX) 及STM32CubeIDE
16	应用程序 开发速度	非常快，有完善的控制逻辑及框架	慢，因为要搭建控制逻辑及框架
17	应用程序 开发难度	非常容易，只要配置好功能，做些相 应数据处理即可完成应用程序	有点困难，因为要搭建控制逻辑及框架
18	应用程序 稳定性	很稳定，软件框架经过多年实践验证， 需要技术人员编写的代码并不多，对编 程人员要求比较低	稳定性依赖编程人员的技术水平
19	适用用户	选择使用中嵌凌云工控板及工控模块， 想快速稳定开发产品，但软件人员技术 水平一般的客户	选择使用中嵌凌云工控板、工控模块及 第三方STM32工控板，软件人员技术水平 很高的客户
20	针对初学者	需要对我们的软件架构进行相应学习， 参考资料及技术支持只能由中嵌凌云来 提供	也要学习软件框架，但网上的参考资料 比较多，技术支持也更丰富一些
21	最新例程命名	含操作系统： AMKNxxxx_MDK_OS_V1.40.00_20260701 不含操作系统： AMKNxxxx_MDK_V1.40.00_20260701	不含操作系统： STM32Cube_FW_F1_V1.8.0-20230627 STM32Cube_FW_F4_V1.25.0-20230627 STM32Cube_FW_H7_V1.25.0-20260106

第二章. AMKN 软件框架例程基本操作

注：以下所述都以 AMKNxxxx_MDK_OS_V1.40.00_20260701 为参考，选择 FreeRTOS 为操作系统，以 AMKN8639 工控板为例说明

1. 打开软件

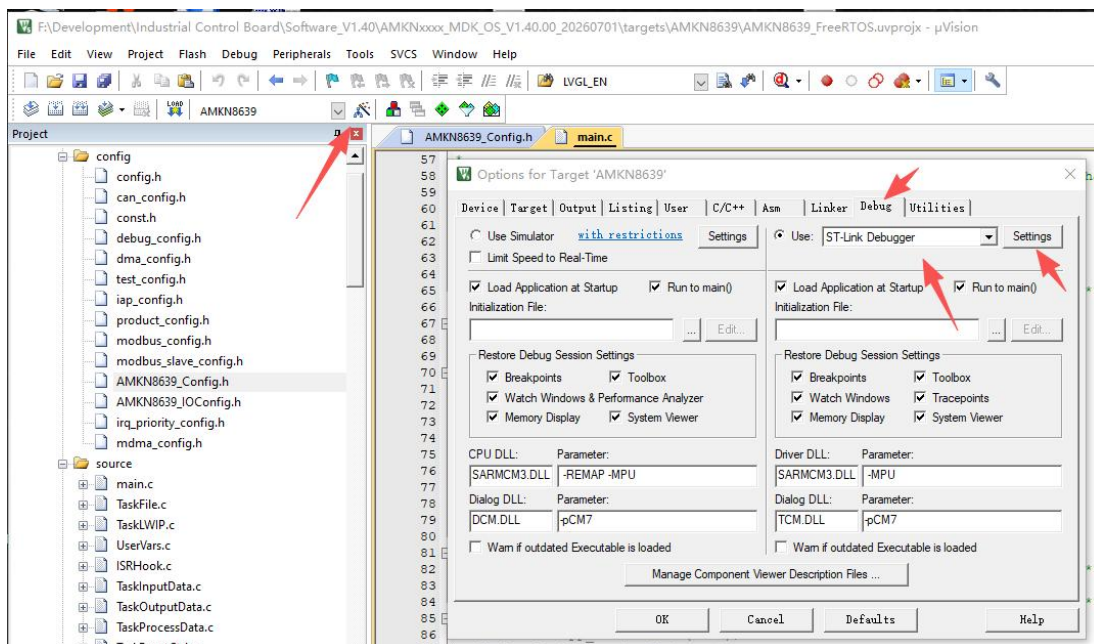
电脑已经安装 Keil (MDK V5.29)，打开 AMKNxxxx_MDK_OS_V1.40.00_20260701\targets\AMKN8639 文件夹，选择打开 AMKN8639_FreeRTOS.uvprojx 含 FreeRTOS 操作系统工程文件，界面如下：



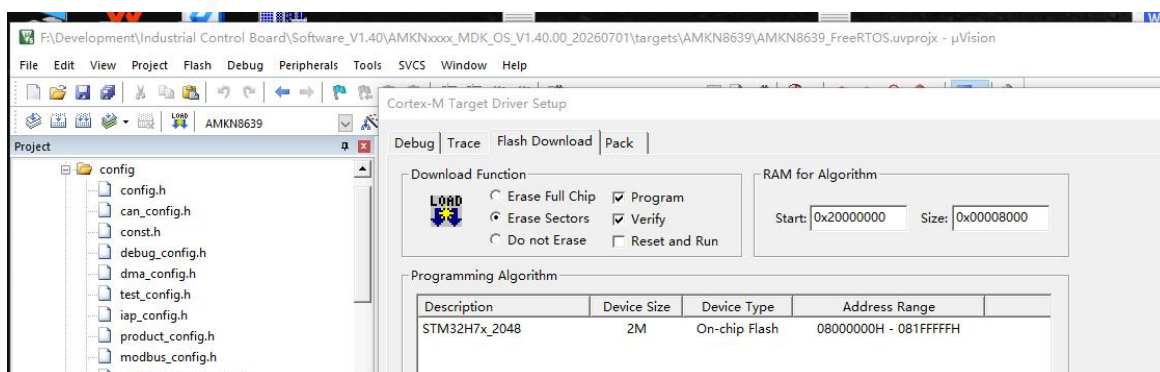
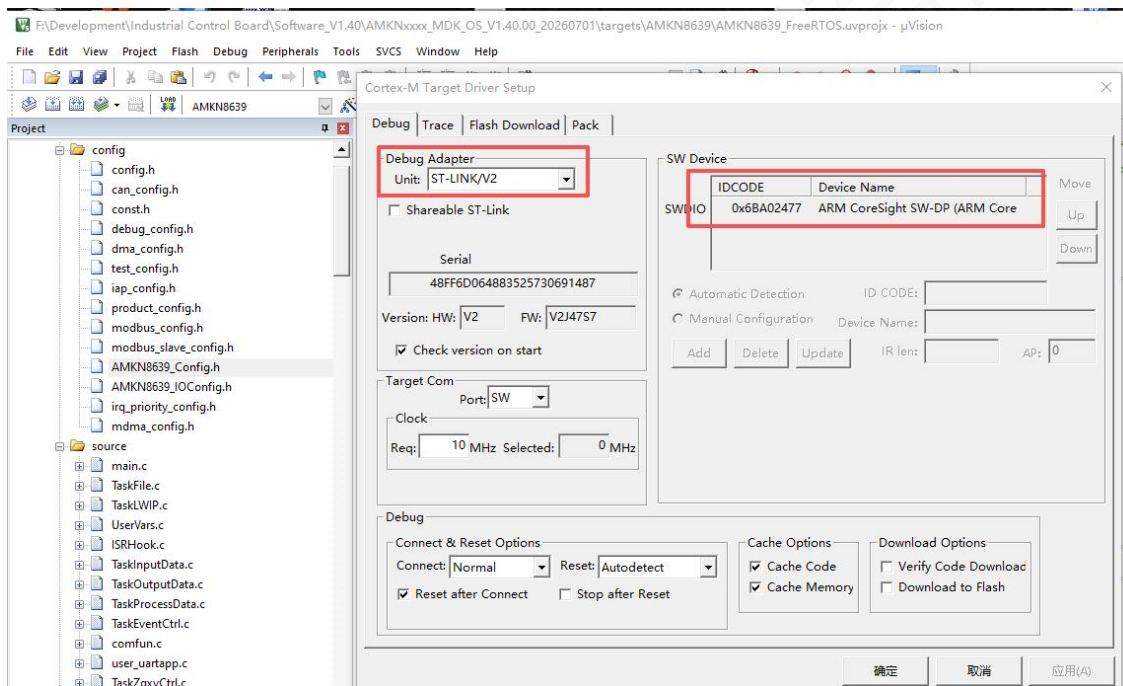
2. 插入 ST-Link 仿真器并给板子加电，如下图：

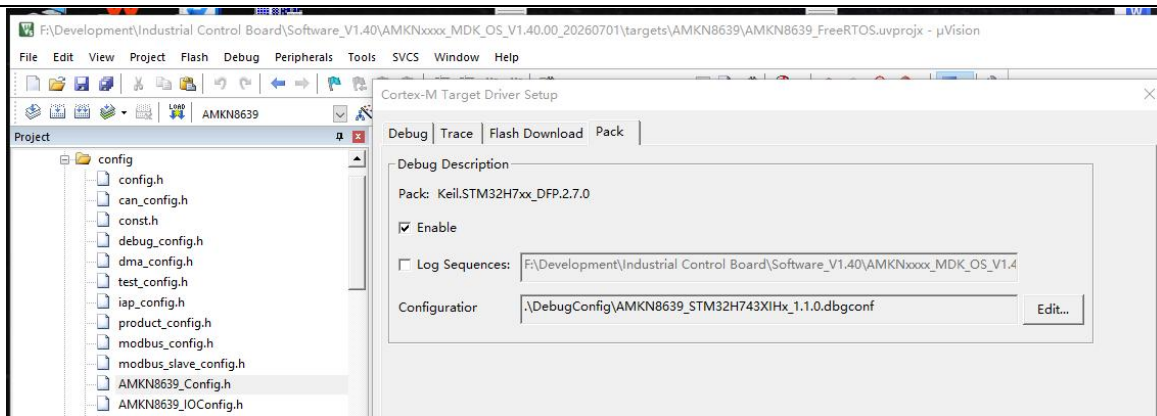


3. 配置 ST-Link 仿真器，如下图：

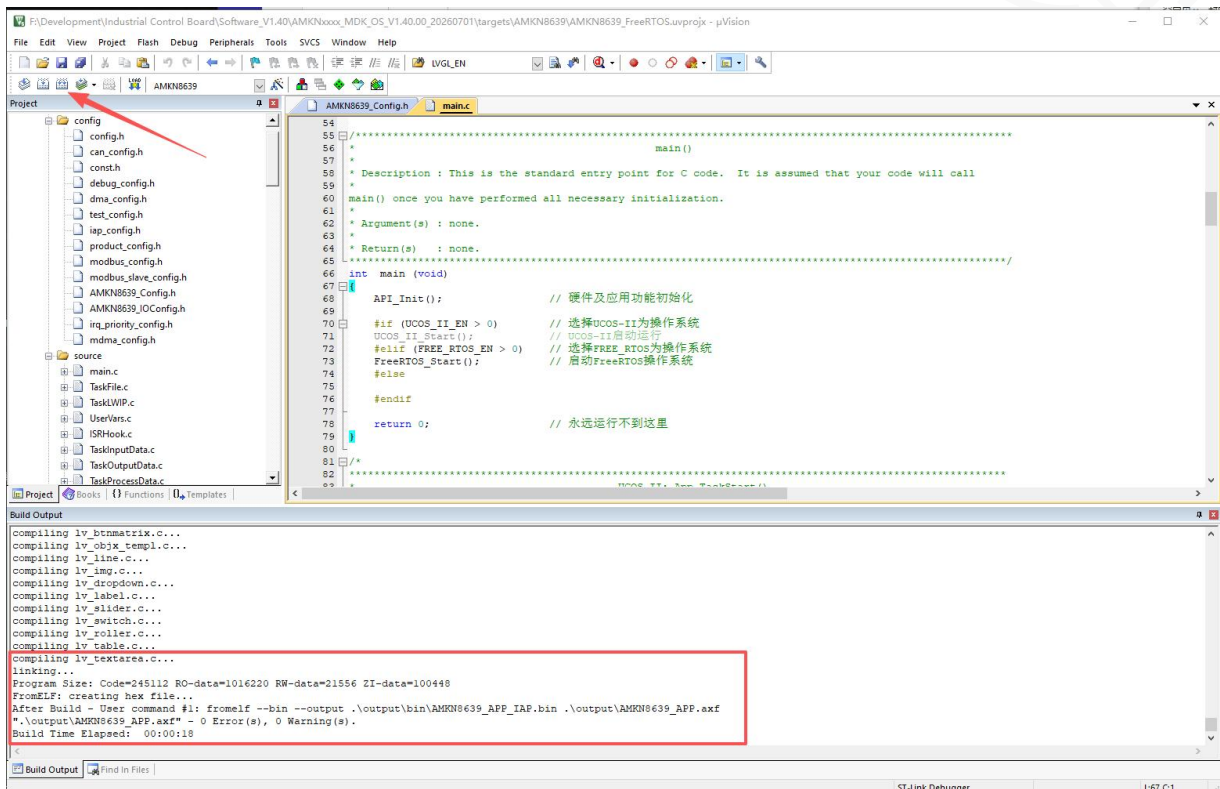


点 Settings 按钮，显示如下，表示已经连接工控板成功

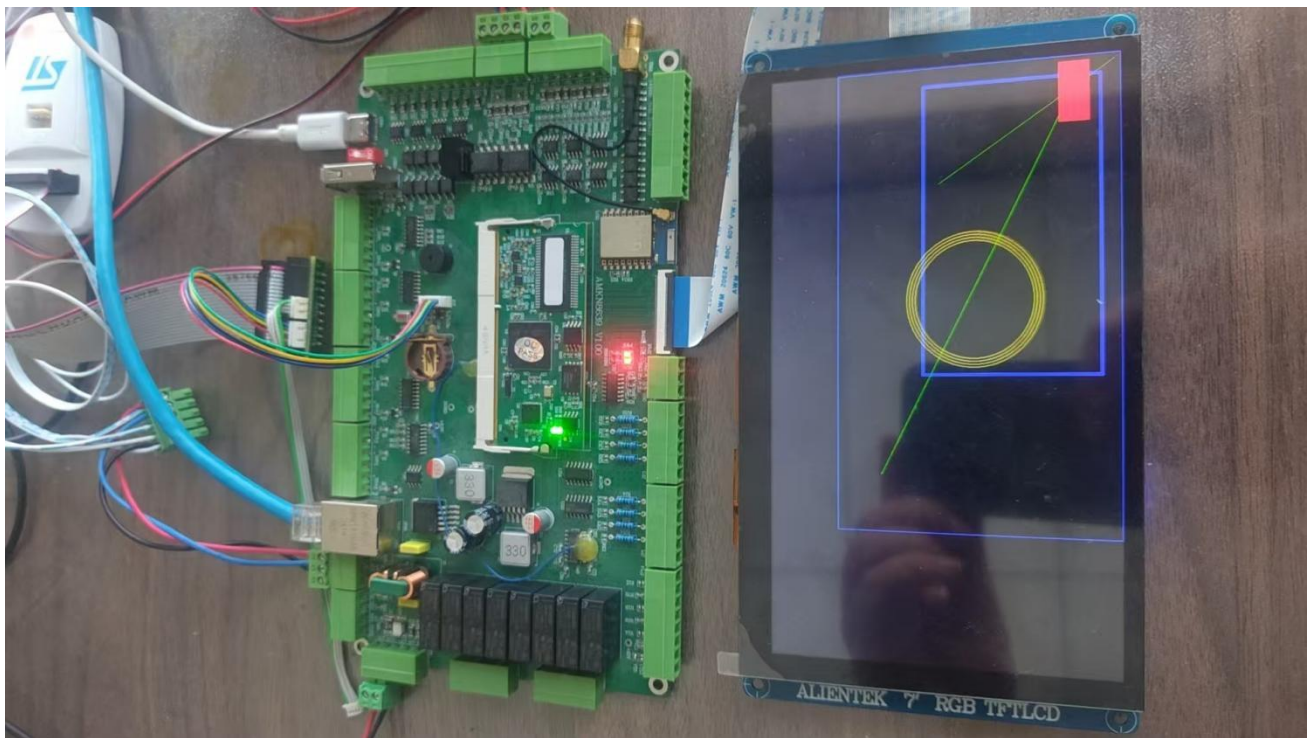




4. 编译软件并下载程序如下图：



5. 点击全速运行，观察板子运行情况



第三章. AMKN 软件框架例程测试功能说明

注：以下所述都以 AMKNxxxx_MDK_OS_V1.40.00_20260701 为参考，选择 FreeRTOS 为操作系统，以 AMKN8639 工控板为例说明

1. RUN LED 运行状态指示

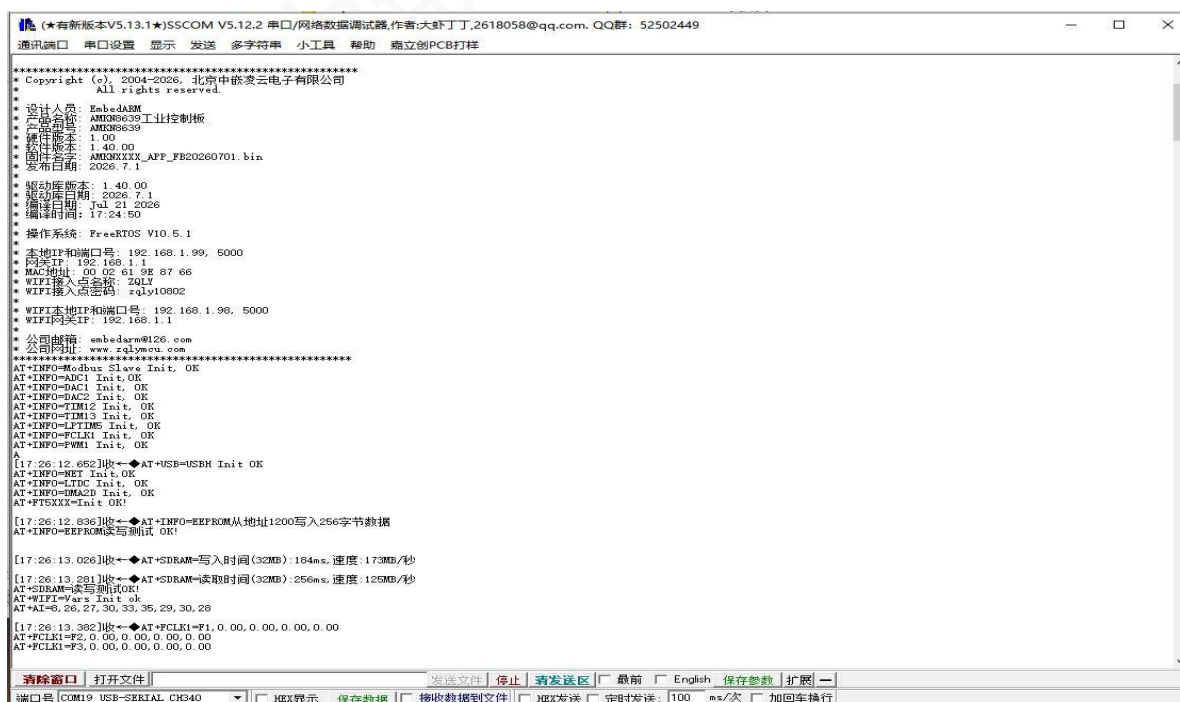
- 1.1 状态：间隔 0.5 秒闪烁，指示程序运行正常；间隔 5 秒，100ms 间隔快闪 5 次；
- 1.2 打开 config/AMKN8639_IOConfig.h 查看 RUN_LED 的端口定义，在这里也可以定义其它 LED
- 1.3 打开 config/test_config.h 查看 APP_RUN_LED_TEST_EN 定义，可以选择关闭 LED 闪烁
- 1.4 打开 source/user_app.c，找到 User_MainAppInit 和 User_MainApp 函数，可以修改下面函数参数 ALED_EventCtrl(RUN_LED_ID, LIBAPP_ALED_CMD_NEG_NUM, xx, xx, xx)，改变 LED 闪烁状态

2. ALARM 蜂鸣器控制

- 2.1 状态：开机蜂鸣器响 200ms，之后间隔 5 秒蜂鸣器响 APP_ALARM_ON_T(20ms) 毫秒；
- 2.2 打开 config/AMKN8639_IOConfig.h 查看 ALARM 的端口定义
- 2.3 打开 config/test_config.h 查看 APP_ALARM_TEST_EN 定义，可以选择关闭蜂鸣器报警
- 2.4 打开 source/user_app.c，找到 User_MainAppInit 和 User_MainApp 函数，可以修改下面函数参数 ALED_EventCtrl(ALARM_ID, LIBAPP_ALED_CMD_ON_T, xx, 0, 0)，改变蜂鸣器报警时间

3. 调试信息输出

- 3.1 习惯使用调试信息输出是编写程序的有效观察手段，必须掌握。
- 3.2 打开 config/debug_config.h 查看，建议必须使能 DEBUG_EN。定义 DEBUG_UART 选择信息输出串口，按默认就行，当然也可以修改。除了 AMKN8804，建议用 UART1 作为调试信息输出端口。
- 3.3 每种工控板 UART1 对外连接端口有可能是 RS232、RS485 或者 USB(USB 转 UART 芯片)，AMKN8639 采用的就是 USB 转 UART 的 USB 接口作为调试输出。
- 3.4 将板子的 USB1 (TYPE C) 接口通过 USB 线连接计算机，计算机需要安装 CH340 的 USB 驱动。计算机上会产生一个 COMx 口。打开串口调试助手，选择 COMx，设置波特率 115200、8 位数据、1 个停止位、无校验，打开串口就可以观察打印输出数据，如下图显示：



```
***** (★有新版本V5.13.1★)SSCOM V5.12.2 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com, QQ群: 52502449 *****
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 断立创PCB打样

*****
* Copyright (c), 2004-2026, 北京中嵌凌云电子有限公司
* All rights reserved.
*
* 设计人员: Ebedarm
* 产品名称: AMKN8639工控板
* 产品型号: AMKN8639
* 版本号: 1.00
* 软件版本: 1.40.00
* 固件名字: AMKN8639_APP_F20260701.bin
* 发布日期: 2026.7.1
*
* 驱动版本: 1.40.00
* 驱动发布日期: 2026.7.1
* 编译日期: Jul 21 2026
* 编译时间: 17:24:50
*
* 操作系统: FreeRTOS V10.5.1
*
* 本地IP和端口号: 192.168.1.99, 5000
* 网关IP: 192.168.1.1
* MAC地址: 00 02 61 9E 87 66
* WIFI接入点名称: ZQLY
* WIFI接入点密码: zqly10802
*
* WIFI本地IP和端口号: 192.168.1.98, 5000
* WIFI网关IP: 192.168.1.1
*
* 公司邮箱: embedarm@126.com
* 公司网址: www.zqlymcu.com
*****
AT+INFO=Modbus Slave Init. OK
AT+INFO=ADC1 Init. OK
AT+INFO=DAC1 Init. OK
AT+INFO=DAC2 Init. OK
AT+INFO=TIMER1 Init. OK
AT+INFO=TIMER2 Init. OK
AT+INFO=TIMER3 Init. OK
AT+INFO=LPTIMER Init. OK
AT+INFO=FLK1 Init. OK
AT+INFO=FLK2 Init. OK
[17:26:12.652]接收 AT+USB=USB Init OK
AT+INFO=EEPROM Init. OK
AT+INFO=EEPROM从地址1200写入256字节数据
AT+INFO=EEPROM读写测试 OK!
[17:26:13.026]接收 AT+SDRAM=写入时间(32MB):184ms,速度:173MB/秒
[17:26:13.281]接收 AT+SDRAM=读取时间(32MB):256ms,速度:125MB/秒
AT+SDRAM=读写测试OK!
AT+WIFI=Yes Init ok
AT+AI=6,26,27,30,33,35,29,30,28
[17:26:13.382]接收 AT+FLK1=F1,0.00,0.00,0.00,0.00
AT+FLK1=F2,0.00,0.00,0.00,0.00
AT+FLK1=F3,0.00,0.00,0.00,0.00

清除窗口 打开文件
端口号 COM19 USB-SERIAL CH340 [X] HEX显示 [X] 保存数据 [X] 接收数据到文件 [X] HEX发送 [X] 定时发送: 100 ms/次 [X] 加回车换行
```

3.5 调试输出信息一般采用 AT 指令形式，具体参考文件《AMKNxxxx 系列工控板(模块)AT 指令_V1.10_20260721.PDF》

3.6 配置文件 config/debug_config.h 也定义了很多常用功能是否开启信息调试输出，用户可以在这里自由设置。

4. DI 端口测试

- 4.1 测试功能：利用串口调试助手软件显示 DI 状态变化。
- 4.2 打开 config/AMKN8639_IOConfig.h 查看 DI 端口定义(不可修改)
- 4.3 打开 config/AMKN8639_Config.h 查看和修改 DI 输入参数配置
- 4.4 打开 config/test_config.h 查看 DI 测试使能，将 APP_DI_TEST_EN 设置为 0，不要设置成 1。
- 4.5 将 JP6 的 COM1 端口接入 24V 电源正极，将 DI1-DI8 分别接入 24V 电源负极(GND)，观察串口调试助手显示 DI 状态如下图：

```
(★有新版本V5.13.1★)SSCOM V5.12.2 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com, QQ群: 52502449
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 嘉立创PCB打样
AT+FCLK1=F3,0.00,0.00,0.00,0.00
[14:50:42.657]收←◆AT+AI=8,26,28,29,32,34,29,31,30
AT+PWM1=F,2000
AT+PWM2=F,2000
AT+PWM3=F,2000
AT+PWM4=F,2000
[14:50:42.776]收←◆AT+INFO=App_TaskLCD
[14:50:42.846]收←◆AT+INFO=TIM13定时器中断
[14:50:42.944]收←◆AT+INFO=IdleCnt:552511, IdleRatio:99
[14:50:42.990]收←◆AT+DI1=OFF
AT+DI=8,00000000
[14:50:43.137]收←◆AT+CAN1=RX[1,8]:a0 08 03 04 05 06 07 08
AT+CAN1=TX[1,8]:a0 08 03 04 05 06 07 08
[14:50:43.243]收←◆AT+INFO=User_MainApp!
AT+DI1=ON
AT+DI=8,10000000
[14:50:43.363]收←◆AT+A01=1997
AT+A02=1997
AT+RTC=2023.4.30,23:59:43,228
AT+PWM1=DIR,1
AT+PWM1=ENA,1
```

AT+DI1=OFF, 表示 DI1 没有加载信号; AT+DI1=ON, 表示 DI1 加载 24V 电压信号;

AT+DI=8,10000000, 表示 8 个 DI 输入, 只有 DI1 加载 24V 电压信号。

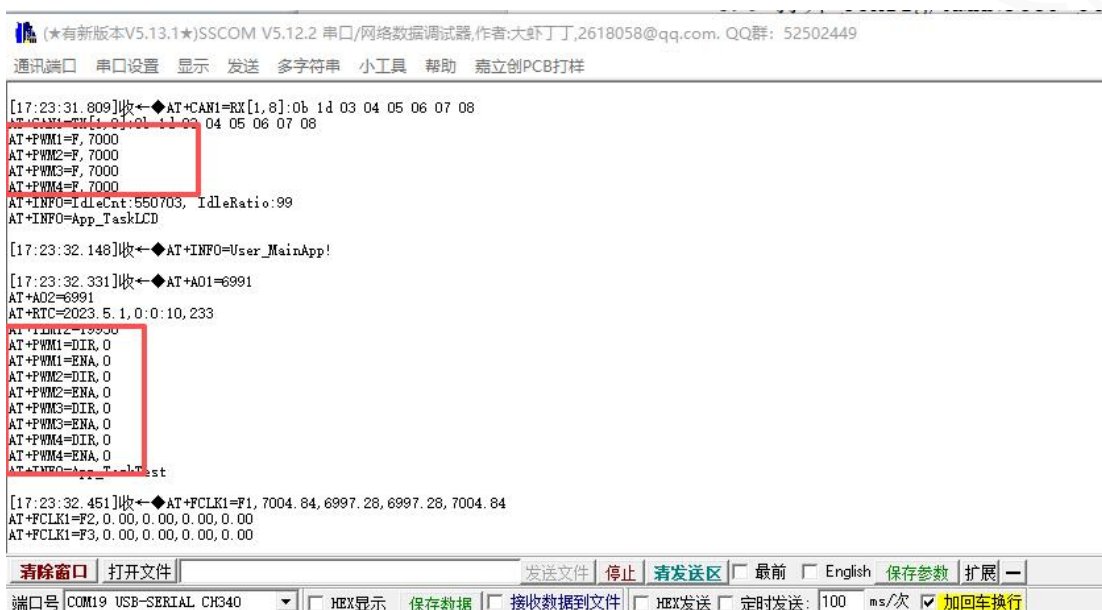
- 4.6 打开 source/user_ioapp.c 查看 DI_DataProcess 和 User_AppDIProc 函数：在这里处理接收 DI 信号变化及应用处理。

5. DO 端口测试

- 5.1 测试功能：间隔 1 秒所有 DO 输出 1 或 0。
- 5.2 打开 config/AMKN8639_IOConfig.h 查看 DO 端口定义(不可修改)
- 5.3 打开 config/AMKN8639_Config.h 查看和修改 DO 输出参数配置
- 5.4 打开 config/test_config.h 查看 DO 测试使能：将 APP_DO_TEST_EN 设置为 1 使能 DO 输出测试。
- 5.5 将 JP5 的+OPT 端口接入 24V 电源正极, 将 IGND 端口接入 24V 电源负极(GND), 用万用表测量 DO1-DO8 的高低电平变化, 来判断 DO 间隔 1 秒输出 1 或 0。
注意：因为有些板子 DO 输出没有加上拉电阻, 万用表测量不出电压变化, 可以在+OPT 和 DOx 之间外接一个 10K 直插电阻, 或者直接在+24V 电源正和 DOx 之间直接加负载(电磁阀或继电器)测试。
- 5.6 JP17 和 JP18 是 DO9-DO16 控制继电器输出接口, 你可以听到继电器动作的声音。用万用表电阻通断档可以测量 COMA 和 K1-K4 之间的通断, COMB 和 K5-K8 之间的通断
- 5.7 打开 source/user_testapp.c 查看 User_AppTest 和 DO_AppTest 函数：控制 DO 间隔 1 秒输出 1 或 0

6. PWM 输出测试

- 6.1 测试功能：在 PWM_FREQ 连续脉冲频率输出模式下：启动 PWM 输出 1khz->2khz->3khz->4khz->5khz->6khz->7khz->8khz->9khz->10khz->停止。每个阶段持续 1 秒，整个过程完成后，重新开始，周而复始循环。
- 6.2 打开 config/AMKN8639_IOConfig.h 查看 PWM 端口定义(不可修改)
- 6.3 打开 config/AMKN8639_Config.h 查看和修改 PWM 输出参数配置：PWM 模式设置为 PWM_FREQ，其它按默认配置。
- 6.4 打开 config/test_config.h 查看 PWM 测试使能：将 APP_PWM_TEST_EN 设置为 1，其它按默认配置。
- 6.5 用示波器测量 JP4、JP8、JP12、JP13 的 PUL+或 PUL-，观察 PWM 信号间隔 1 秒频率变化。
用示波器测量 JP4、JP8、JP12、JP13 的 DIR+或 DIR-，观察 DIR 信号间隔 1 秒高低电平变化。
用示波器测量 JP4、JP8、JP12、JP13 的 ENA+，固定是 5V 电压。
用示波器测量 JP4、JP8、JP12、JP13 的 ENA-，电压 0V，因为是 NPN 开路输出。
- 6.6 观察串口调试助手显示 PWM 输出频率及 DIR 和 ENA 输出值，如下图：



- 6.7 打开 source/user_testapp.c 查看 User_AppTest 中的 PWM 间隔 100ms 处理测试部分程序调用 PWM_AppTest 和 PWM_AppTest_DirEnaCtrl 函数
- 打开 source/user_pwmapp.c 查看 PWM AppTest 和 PWM AppTest_DirEnaCtrl 函数

7. FCLK 频率测量

- 7.1 测试功能：利用串口调试助手软件显示 FCLK 输入脉冲频率。
- 7.2 打开 config/AMKN8639_IOConfig.h 查看 FCLK 端口定义(不可修改)
- 7.3 打开 config/AMKN8639_Config.h 查看和修改 FCLK 参数配置：模式选择测频模式，其它默认。
- 7.4 打开 config/test_config.h 查看 FCLK 测试使能，将 APP_DI_TEST_EN 设置为 1，不要设置成 0。
- 7.5 将 JP14 的 FA+/FA-、FB+/FB-、FZ+/FZ-差分输入分别连接 JP4 的 PUL+/PUL-，就是将 PWM 输出信号接入 FCLK 输入，观察串口调试助手显示 FCLK 测量的频率，如下图：



AT+FCLK1=F1,3000.22,... 表示 FCLK1 通道 CH1 (FA+/FA-) 测量频率是 3000.22HZ

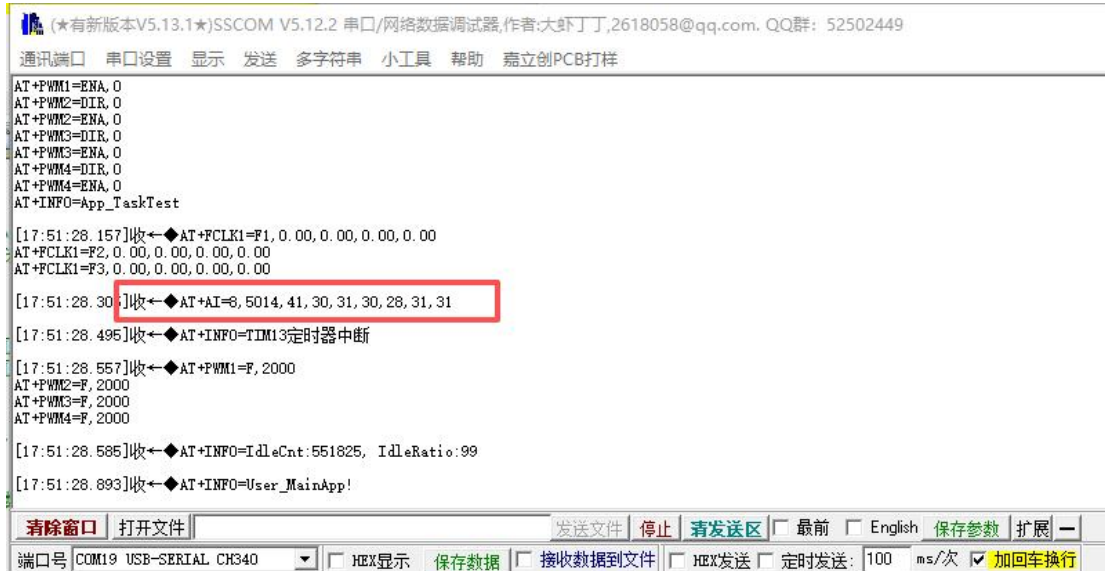
AT+FCLK1=F2,0.00,... 表示 FCLK1 通道 CH2 (FB+/FB-) 测量频率是 0HZ

AT+FCLK1=F3,0.00,... 表示 FCLK1 通道 CH3 (FZ+/FZ-) 测量频率是 0HZ

- 7.6 打开 source/user_flcapp.c 查看 FCLK_DataProcess 和 FCLKx_UserProcess 函数：在这里处理接收 FCLK 测量值及应用处理。

8. AI (ADC) 端口输入测试

- 8.1 测试功能：利用串口调试助手软件显示 AI 端输入信号的电压值。
- 8.2 打开 config/AMKN8639_IOConfig.h 查看 AI 端口定义(不可修改)
- 8.3 打开 config/AMKN8639_Config.h 查看和修改 ADC 参数配置：按默认配置。
- 8.4 将 JP1 的 AI1、AI2、AI3、AI4 和 JP3 的 AI5、AI6、AI7、AI8 分别接入+5V 电压(可以用 JP2 的+5V 输出端口)，观察串口调试助手显示 AI 测量值, 如下图：



AT+AI=8, 5014, 41, 30, 31, 30, 28, 31, 31: 8 表示 8 个通道，5014 表示 AI1 输入电压是 5.014V，41 表示 AI2 输入电压是 0.041V，以此类推后面依次表示 AI3、AI3、AI4、AI5、AI6、AI7、AI8 的输入电压值

- 8.5 打开 source/user_adcapp.c 查看 ADC_DataProcess 函数：在这里处理接收 AI 测量值及应用处理。

9. AO (DAC) 端口输出测试

- 9.1 测试功能：间隔 1 秒将 DAC 输出值按 0V, 1V, 2V, 3V, 4V, 5V, 6V, 7V, 8V, 9V, 10V 循环变化。
- 9.2 打开 config/AMKN8639_Config.h 查看和修改 DAC 参数配置：设置为手动输出模式，其它默认。
- 9.3 打开 config/test_config.h 查看 DAC 测试使能，将 APP_DAC_TEST_EN 设置为 1，不要设置成 0。
- 9.4 用万用表测量 JP11 的 A01 和 A02 的电压变化：间隔 1 秒将 DAC 输出值按 0V, 1V, 2V, 3V, 4V, 5V, 6V, 7V, 8V, 9V, 10V 循环变化。观察串口调试助手显示 AO 输出值，如下图：



AT+A01=8989：表示 A01 输出电压是 8.989V

AT+A02=8989：表示 A03 输出电压是 8.989V

- 9.5 打开 source/user_testapp.c 查看 User_AppTest 中的间隔 1 秒调用 DAC_AppTest
打开 libapp/dac_app.c 查看 DAC_AppTest 和 DAC_AppMTOutTest 函数

10. RS232 和 RS485 通信端口测试

10.1 测试功能：所有 RS232 和 RS485 的通信端口，向该端口发送任何数据，都会原数据返回。

10.2 默认配置是波特率 115200、8 位数据、1 个停止位、无校验

10.3 打开 config/AMKN8639_IOConfig.h 查看 UART(RS232 和 RS485)的 IO 端口定义(不可修改)

10.4 打开 config/AMKN8639_Config.h 查看和修改 UART1-UARTx(RS232 和 RS485)的功能配置定义

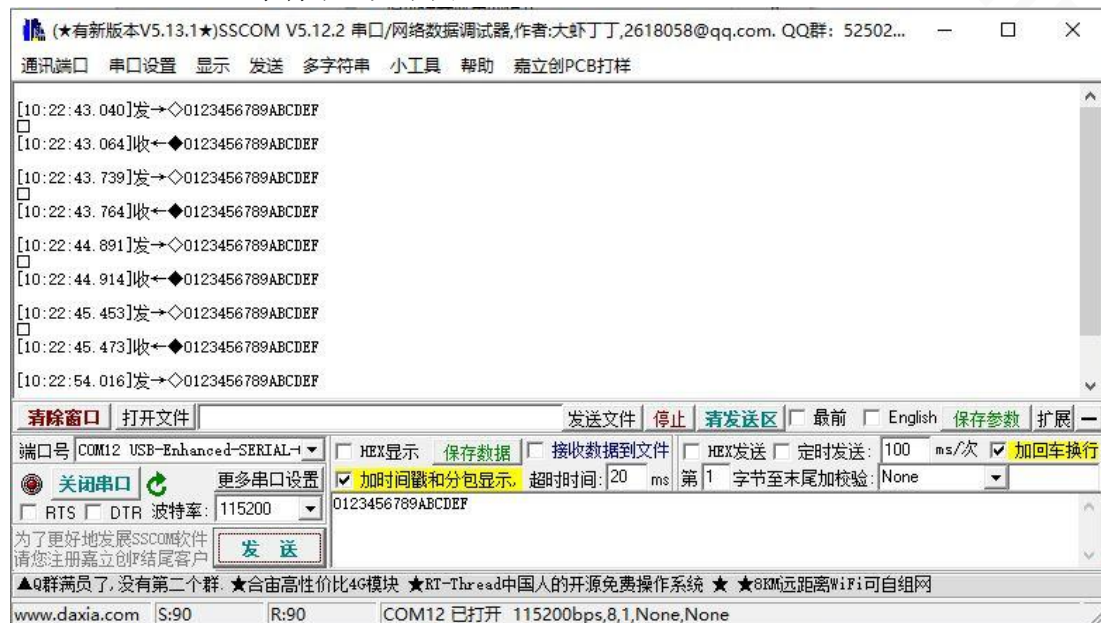
10.5 打开 config/test_config.h 查看 UART 测试使能配置，可以关闭和打开该测试功能，也可以使能自动发送数据测试。

10.6 将 JP9 和 JP20 的 2 个 RS232 接口连接到计算机的 USB 转 RS232 接口注意要交叉连接：

TX<->RX, RX<->TX, DGND<->GND。

将 JP7 的 3 个 RS485 接口连接到计算机的 USB 转 RS485 接口注意要顺序连接：Ax+<->A, Bx-<->B。

利用串口调试助手测试，向端口发送字符串 0123456789ABCDEF，则会接收到 0123456789ABCDEF 字符串，如下图：

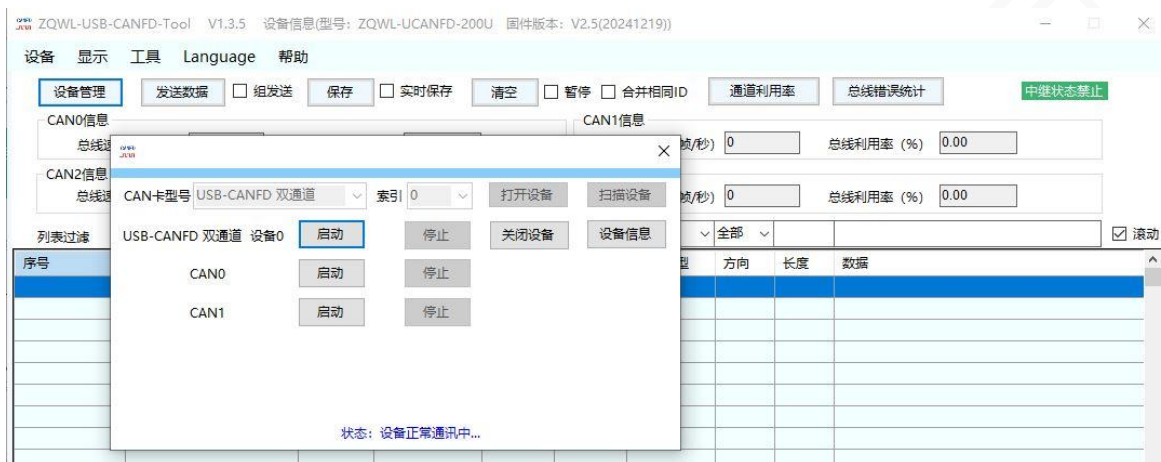


10.7 打开 source/user_uartapp.c 查看 Uartx_UserProcess 函数：实现接收数据后再原数据发出

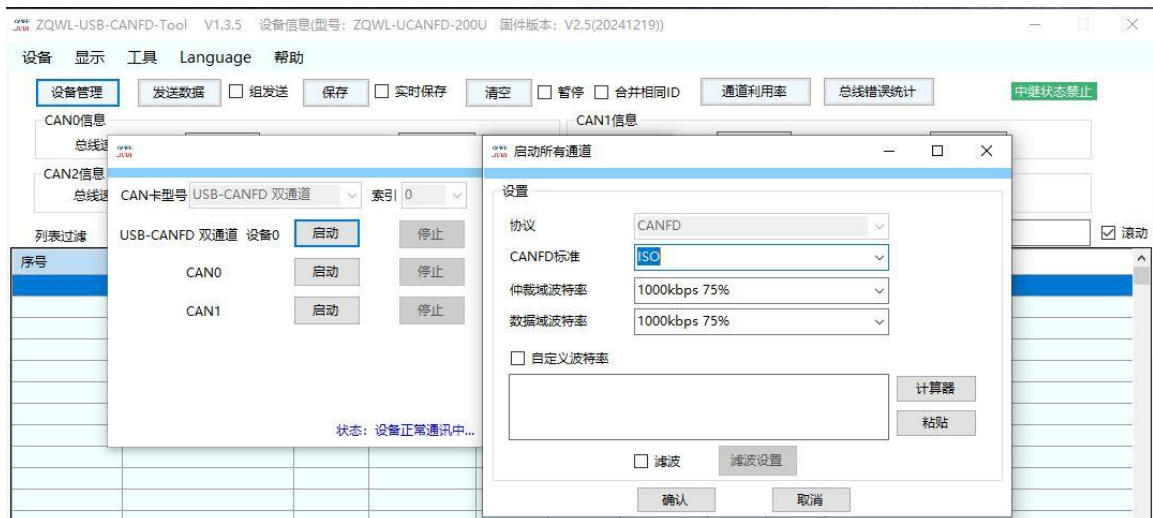
10.8 打开 source/user_testapp.c 查看 User_AppTest 和 Uart_AppTxTest 函数：实现自动发送数据(当 APP_UARTx_TEST_EN 和 APP_UARTx_TEST_TX 使能时)

11. CAN 通信端口测试

- 11.1 测试功能：所有 CAN 通信端口，向该端口发送任何数据，都会原数据返回。
- 11.2 默认配置是波特率 1MHZ、标准 CAN 模式、扩展帧模式、范围滤波器：1-15
- 11.3 打开 config/AMKN8639_IOConfig.h 查看 CAN 的 IO 端口定义(不可修改)
- 11.4 打开 config/AMKN8639_Config.h 查看和修改 CAN1/CAN2 的功能配置定义
- 11.5 打开 config/can_config.h 查看和修改 CAN1/CAN2 的滤波器配置定义
- 11.6 打开 config/test_config.h 查看 CAN 测试使能配置，可以关闭和打开该测试功能，也可以使能自动发送数据测试。
- 11.7 将 JP19 的 2 个 CAN 接口连接到计算机的 USB 转 CAN 设备上，注意连接方式：CANxH<->CANx_H, CANxL<->CANx_L。设备型号：ZQWL-UCANFD-200U
打开 ZQWL-USB-CANFD-Tool 工具软件进行测试，点击设备管理，如下图：



点击启动后如下图：



按图设置，再点确认关闭波特率设置，再关闭设备管理窗口，完成配置。

点击发送数据，如下图设置，点击发送按钮，启动发送测试，会看到发送数据和接收数据相同：

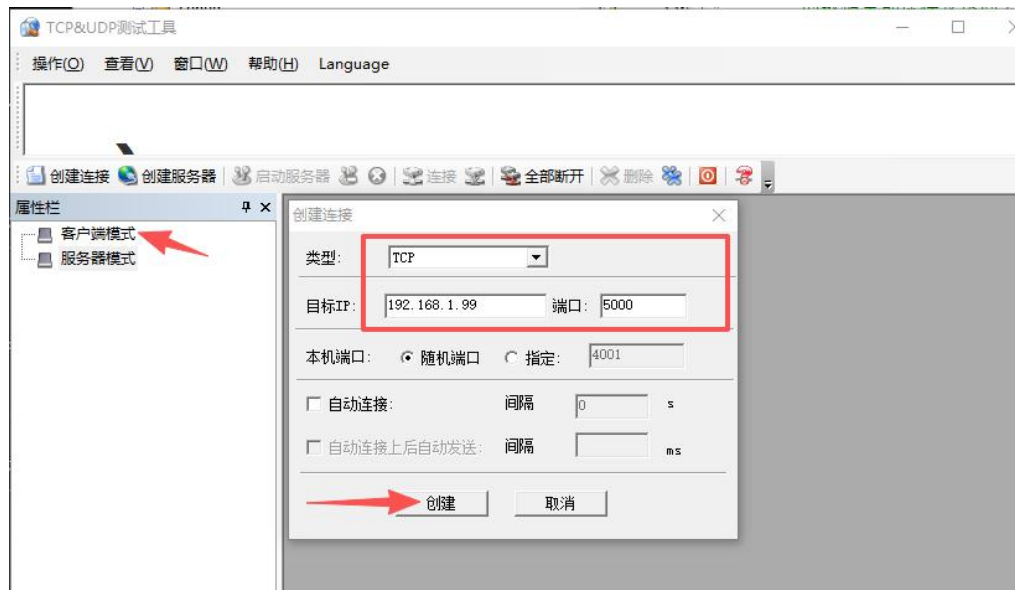


11.8 打开 source/user_canapp.c 查看 CANx_UserProcess 函数：实现接收数据后再原数据发出

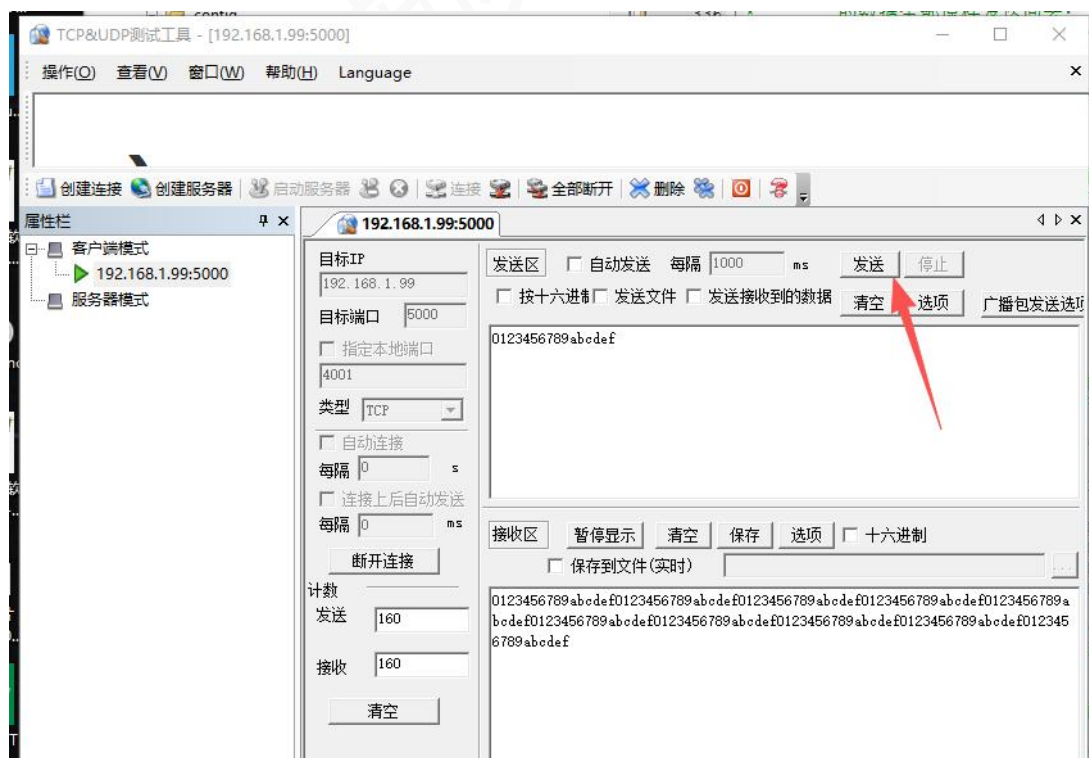
11.9 打开 source/user_testapp.c 查看 User_AppTest 和 CAN_AppTxTest 函数：实现自动发送数据(当 APP_CANx_TEST_EN 和 APP_CANx_TEST_TX 使能时)

12. 网络通信测试

- 12.1 测试功能：板子作为服务端，客户端软件连接成功后，向其发送任何数据，都会原数据返回。
- 12.2 默认配置是：板子 IP:192.168.1.99 端口：5000
- 12.3 打开 config/AMKN8639_IOConfig.h 查看网络管脚 IO 定义(不可修改)
- 12.4 打开 config/AMKN8639_Config.h 查看和修改 LWIP 的功能配置：本地 IP：192.168.1.99，本地端口：5000，选择 TCP 服务器模式，LWIP_TCP_SERVER_EN 配置为 1，其它配置默认。
- 12.5 打开 config/test_config.h 查看 NET 网络测试使能配置，可以关闭和打开该测试功能，也可以使能自动发送数据测试。
- 12.6 用网线连接板子网口和计算机，打开网络测试软件，如图：
创建客户端



连接服务器，发送数据测试



- 12.7 打开 source/user_netwifiapp.c 查看 NET_TCPServerDataProcess 函数：实现接收数据后再原数据发出
- 12.8 打开 source/user_testapp.c 查看 User_AppTest 和 NET_UserTCPServer_TxTest 函数：实现自动发送数据(当 APP_NET_TEST_EN 和 APP_NET_TCP_SERVER_TX_TEST_EN 使能时)

13. WIFI 通信测试

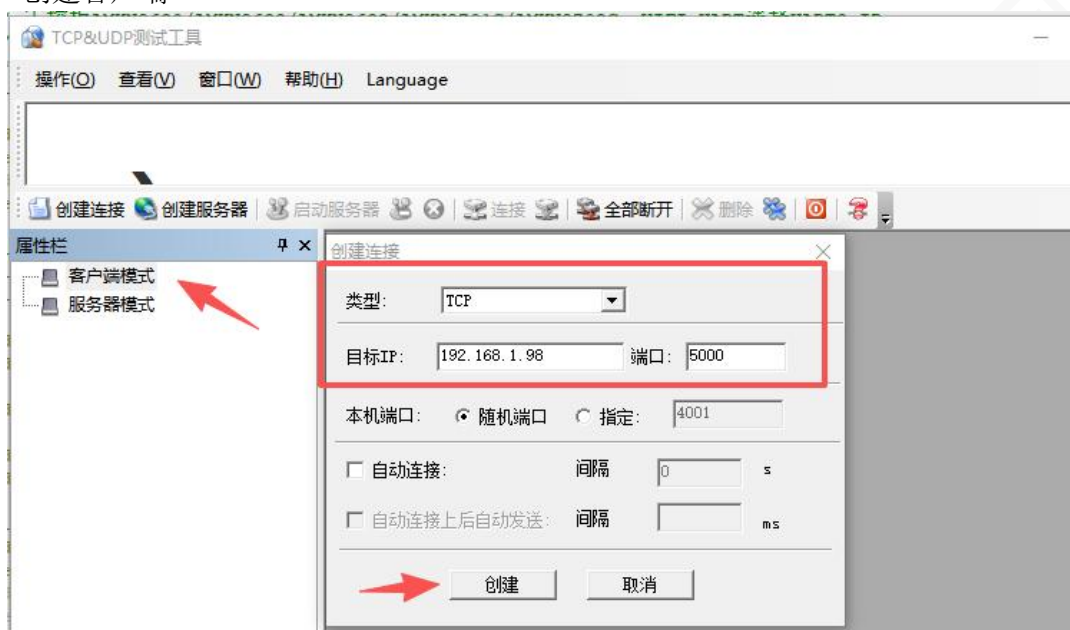
13.1 测试功能：板子 WIFI 模块作为服务端，客户端软件连接成功后，向其发送任何数据，都会原数据返回。

13.2 默认 WIFI 模块配置是：IP:192.168.1.98 端口：5000

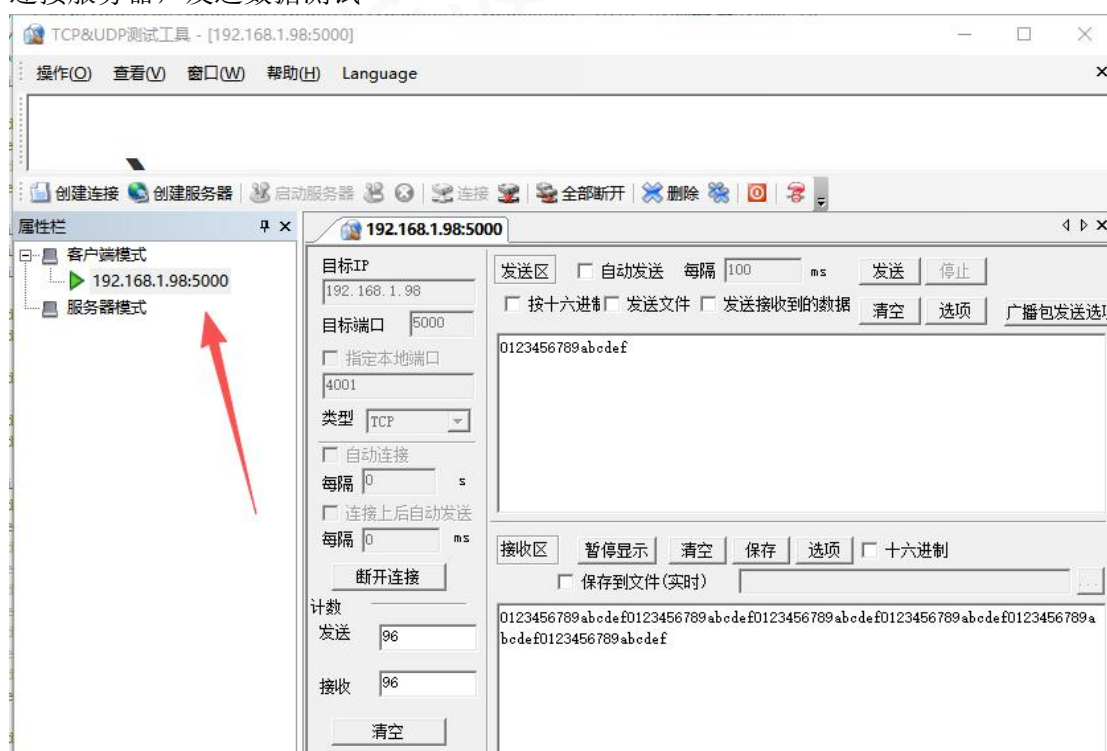
13.3 打开 device/wifi/wifi_config.h 查看和修改功能配置：设置用户路由器 AP 接入点 WIFI_ROUTER_SSID；设置接入点密码 WIFI_ROUTER_PASSWORD；默认本地 IP (WIFI_LOCAL_IP)，192.168.1.98；本地端口 (WIFI_LOCAL_PORT) 5000；其它按默认配置；
查看下 WIFI_LINK_ID1_EN 使能，LINK_ID1 固定作为 TCP SERVER 模式通信
查看下 APP_WIFI_TEST_EN 使能打开

13.4 用网线连接板子网口和计算机，打开网络测试软件，如图：

创建客户端



连接服务器，发送数据测试



- 13.5 打开 source/user_netwifiapp.c 查看 WIFI_DataProcess 函数：实现接收数据后再原数据发出；查看 WIFI_UserTxProc 函数，发送数据；
- 13.6 打开 source/user_testapp.c 查看 User_AppTest 函数，间隔 1 秒调用 WIFI_UserTxProc 函数：实现自动发送数据(当 APP_WIFI_TEST_EN 和 APP_WIFI_TX_TEST_EN 使能时)

14. EEPROM 读写测试

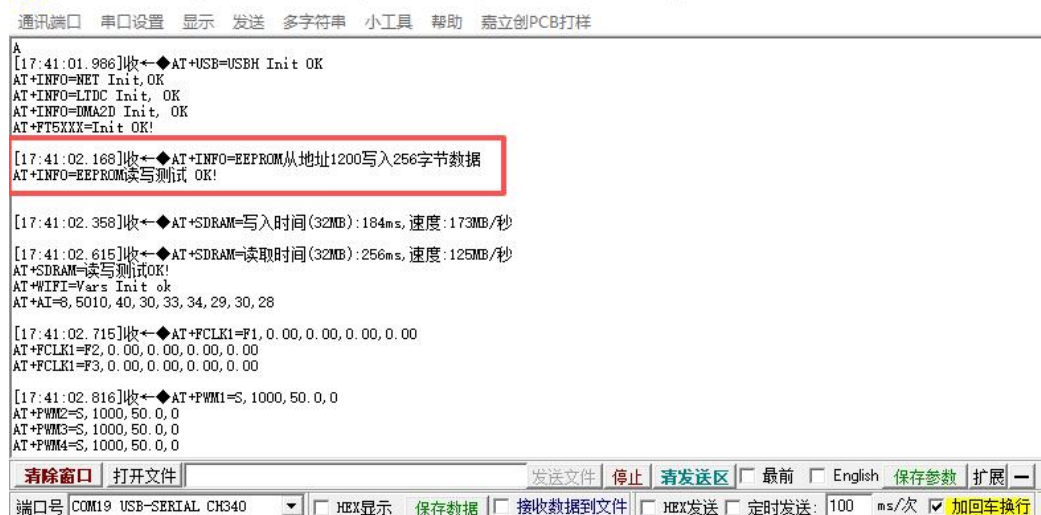
14.1 测试功能：开机自动进行 EEPROM 读写测试。

14.2 打开 config/AMKN8639_IoConfig.h 查看 I2C1(连接 EEPROM)的 IO 端口定义(不可修改)

14.3 打开 config/AMKN8639_Config.h 查看 EEPROM 的功能配置定义, 不要修改

14.4 打开 config/test_config.h 查看 EEPROM 测试使能配置, 可以关闭和打开该测试功能。
可以定义起始地址和读写数据长度

14.5 开机后测试结果显示:



```
A
[17:41:01.986]收←◆AT+USB=USBH Init OK
AT+INFO=NET Init, OK
AT+INFO=LTDC Init, OK
AT+INFO=DMA2D Init, OK
AT+FT5XXX=Init OK!
[17:41:02.168]收←◆AT+INFO=EEPROM从地址1200写入256字节数据
AT+INFO=EEPROM读写测试 OK!
[17:41:02.358]收←◆AT+SDRAM=写入时间(32MB):184ms, 速度:173MB/秒
[17:41:02.615]收←◆AT+SDRAM=读取时间(32MB):256ms, 速度:125MB/秒
AT+SDRAM=读写测试OK!
AT+WIFI=Vers Init ok
AT+AI=8, 5010, 40, 30, 33, 34, 29, 30, 28
[17:41:02.715]收←◆AT+FCLK1=F1, 0.00, 0.00, 0.00, 0.00
AT+FCLK1=F2, 0.00, 0.00, 0.00, 0.00
AT+FCLK1=F3, 0.00, 0.00, 0.00, 0.00
[17:41:02.816]收←◆AT+PWM1=S, 1000, 50.0, 0
AT+PWM2=S, 1000, 50.0, 0
AT+PWM3=S, 1000, 50.0, 0
AT+PWM4=S, 1000, 50.0, 0
```

14.6 打开 source/user_testapp.c 查看 User_AppTestInit 函数, 调用 EEPROM_AppTest 函数: 实现 EEPROM 读写测试(当 APP_EEPROM_TEST_EN 使能时)

14.7 打开 libapp/eeprom_app.c 查看 EEPROM_AppTest 读写测试函数;

15. SPI Flash (25Q64) 读写测试

15.1 测试功能：开机自动进行 SPIFlash 读写测试。

15.2 打开 config/AMKN8639_IOConfig.h 查看 QSPI 管脚定义 (不可修改)

15.3 打开 config/AMKN8639_Config.h 查看 SPI FLASH (W25QXX 系列) 配置定义, 不要修改

15.4 打开 config/test_config.h 查看 SPIFLASH 测试使能配置, 可以关闭和打开该测试功能。
可以定义起始地址和读写数据长度。

15.5 开机后测试结果显示:

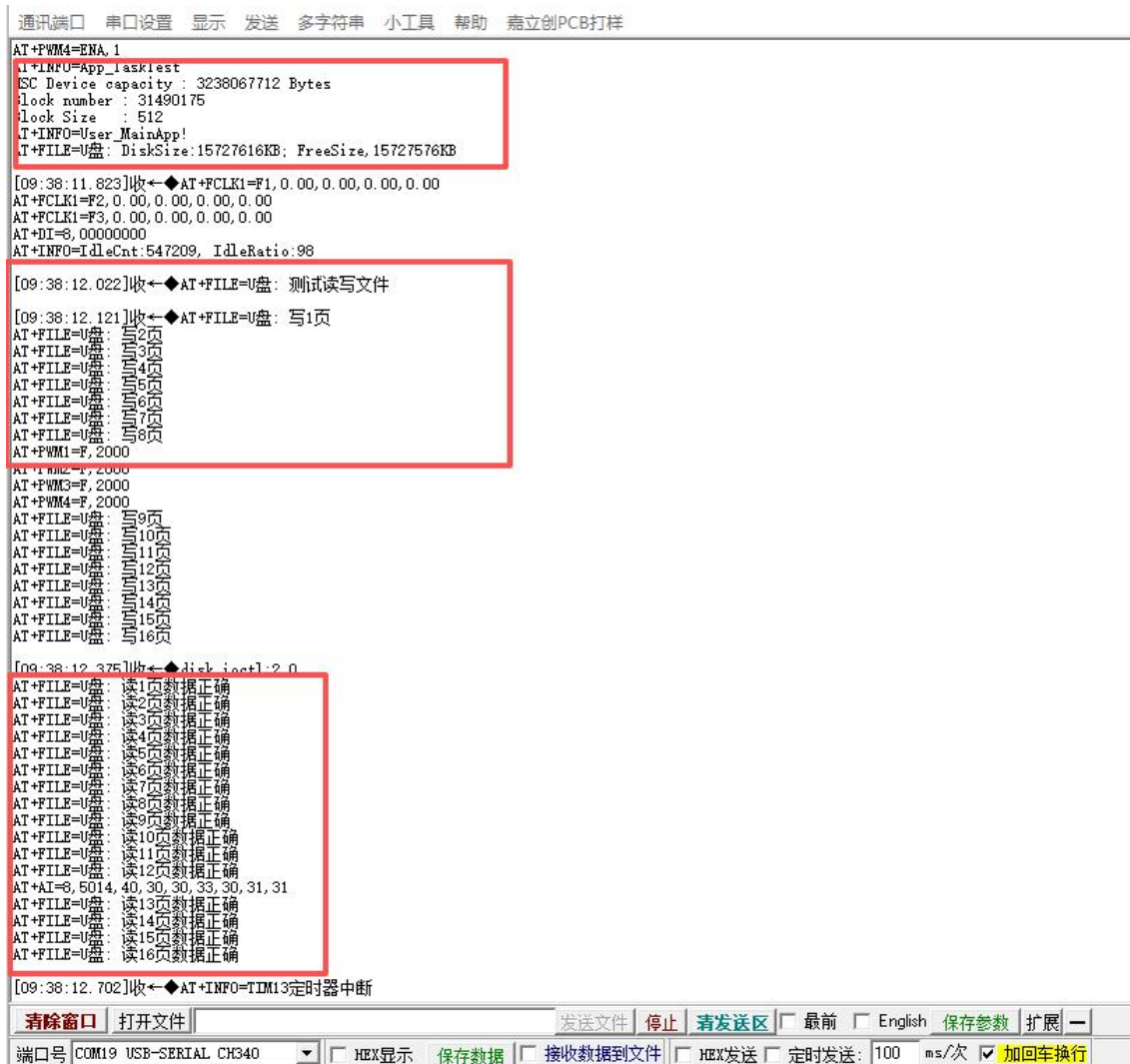
```
AT+PWRM3=F,2000
AT+PWRM4=W,2000
AT+FILE=SPIFlash: 写8页
AT+FILE=SPIFlash: 写9页
AT+FILE=SPIFlash: 写10页
AT+NET=硬件连接,OK
AT+FILE=SPIFlash: 写11页
[08:30:25.156]收←◆AT+FILE=SPIFlash: 写12页
AT+FILE=SPIFlash: 写13页
AT+FILE=SPIFlash: 写14页
AT+FILE=SPIFlash: 写15页
AT+FILE=SPIFlash: 写16页
[08:30:25.232]收←◆disk_ioctl:0,0
[08:30:25.259]收←◆AT+FILE=SPIFlash: 读1页数据正确
AT+FILE=SPIFlash: 读2页数据正确
AT+FILE=SPIFlash: 读3页数据正确
AT+FILE=SPIFlash: 读4页数据正确
AT+FILE=SPIFlash: 读5页数据正确
AT+FILE=SPIFlash: 读6页数据正确
AT+FILE=SPIFlash: 读7页数据正确
AT+FILE=SPIFlash: 读8页数据正确
AT+FILE=SPIFlash: 读9页数据正确
AT+FILE=SPIFlash: 读10页数据正确
AT+FILE=SPIFlash: 读11页数据正确
AT+FILE=SPIFlash: 读12页数据正确
AT+FILE=SPIFlash: 读13页数据正确
AT+FILE=SPIFlash: 读14页数据正确
AT+FILE=SPIFlash: 读15页数据正确
AT+FILE=SPIFlash: 读16页数据正确
```

15.6 打开 source/user_testapp.c 查看 User_AppTestInit 函数, 调用 W25QXX_AppTest 函数: 实现 SPIFlash 读写测试 (当 APP_SPIFLASH_TEST_EN 使能时)

15.7 打开 libapp/spiflash_app.c 查看 W25QXX_AppTest 读写测试函数;

16. U 盘读写测试

- 16.1 测试功能：插入 U 盘自动进行读写测试，注意：U 盘要提前格式化成 FAT32。
- 16.2 打开 config/AMKN8639_IOConfig.h 查看 USB 管脚定义(不可修改)
- 16.3 打开 config/AMKN8639_Config.h 查看 USB 主机模式配置, 不要修改
- 16.4 打开 config/test_config.h 查看 U 盘 File 文件读写测试使能配置, 可以关闭和打开该测试功能。
可以定义读写包大小和数量。
- 16.5 插入 U 盘后测试结果显示：



```
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 嘉立创PCB打样
AT+PWM4=ENA,1
AT+INFO=App_test
ES Device capacity : 3238067712 Bytes
Block number : 31490175
Block Size : 512
AT+INFO=User_MainApp!
AT+FILE=U盘: DiskSize:15727616KB; FreeSize,15727576KB

[09:38:11.823]收←◆AT+FCLK1=F1,0.00,0.00,0.00,0.00
AT+FCLK1=F2,0.00,0.00,0.00,0.00
AT+FCLK1=F3,0.00,0.00,0.00,0.00
AT+DI=8,00000000
AT+INFO=IdleCnt:547209, IdleRatio:98

[09:38:12.022]收←◆AT+FILE=U盘: 测试读写文件

[09:38:12.121]收←◆AT+FILE=U盘: 写1页
AT+FILE=U盘: 写2页
AT+FILE=U盘: 写3页
AT+FILE=U盘: 写4页
AT+FILE=U盘: 写5页
AT+FILE=U盘: 写6页
AT+FILE=U盘: 写7页
AT+FILE=U盘: 写8页
AT+PWM1=F,2000
AT+PWM2=F,2000
AT+PWM3=F,2000
AT+PWM4=F,2000
AT+FILE=U盘: 写9页
AT+FILE=U盘: 写10页
AT+FILE=U盘: 写11页
AT+FILE=U盘: 写12页
AT+FILE=U盘: 写13页
AT+FILE=U盘: 写14页
AT+FILE=U盘: 写15页
AT+FILE=U盘: 写16页

[09:38:12.375]收←◆disk_test1:2,0
AT+FILE=U盘: 读1页数据正确
AT+FILE=U盘: 读2页数据正确
AT+FILE=U盘: 读3页数据正确
AT+FILE=U盘: 读4页数据正确
AT+FILE=U盘: 读5页数据正确
AT+FILE=U盘: 读6页数据正确
AT+FILE=U盘: 读7页数据正确
AT+FILE=U盘: 读8页数据正确
AT+FILE=U盘: 读9页数据正确
AT+FILE=U盘: 读10页数据正确
AT+FILE=U盘: 读11页数据正确
AT+FILE=U盘: 读12页数据正确
AT+AI=8,5014,40,30,30,33,30,31,31
AT+FILE=U盘: 读13页数据正确
AT+FILE=U盘: 读14页数据正确
AT+FILE=U盘: 读15页数据正确
AT+FILE=U盘: 读16页数据正确

[09:38:12.702]收←◆AT+INFO=TIM13定时器中断

清除窗口 打开文件 发送文件 停止 请发送区 最前 English 保存参数 扩展
端口号 COM19 USB-SERIAL CH340 HEX显示 保存数据 接收数据到文件 HEX发送 定时发送: 100 ms/次 加回车换行
```

- 16.6 打开 source/user_testapp.c 查看 User_FileAppTest 函数，调用 APP_FileCtrl_Test 函数：实现 U 盘文件读写测试(当 APP_UDISK_FILE_TEST_EN 使能时，APP_UDISK_FILE_TEST_MODE 是 1)
- 16.7 打开 libapp/file_app.c 查看 APP_FileCtrl_Test 读写测试函数；

17. SD 卡读写测试

17.1 测试功能：

如果 SD 卡是外置插卡：插入 SD 自动进行读写测试；如果是芯片 SD 卡开机自动测试；
注意：插卡 SD 卡要提前格式化成 FAT32。

17.2 打开 config/AMKN8639_IOConfig.h 查看测试功能：开机自动进行 SPIFlash 读写测试。

17.3 打开 config/AMKN8639_IOConfig.h 查看 SD 卡管脚定义(不可修改)

17.4 打开 config/AMKN8639_Config.h 查看 SD 卡配置, 不要修改

17.5 打开 config/test_config.h 查看 SD 卡 File 文件读写测试使能配置，可以关闭和打开该测试功能。可以定义读写包大小和数量。

17.6 插入 SD 后或开机后测试结果显示：



```
AT+FILE=SPIFlash: 读14页数据正确
AT+FILE=SPIFlash: 读15页数据正确
AT+FILE=SPIFlash: 读16页数据正确
AT+FILE=SD卡: 测试读写文件
AT+FILE=SD卡: 写1页
AT+FILE=SD卡: 写2页
AT+FILE=SD卡: 写3页
AT+INFO=TIMER3定时器中断
AT+FILE=SD卡: 写4页

[09:59:34.525]收←AT+INFO=User_MainApp!
AT+FILE=SD卡: 写5页
AT+FILE=SD卡: 写6页
AT+FILE=SD卡: 写7页
AT+FILE=SD卡: 写8页
AT+FILE=SD卡: 写9页
AT+FILE=SD卡: 写10页
AT+FILE=SD卡: 写11页
AT+FILE=SD卡: 写12页
AT+FILE=SD卡: 写13页
AT+FILE=SD卡: 写14页
AT+FILE=SD卡: 写15页
AT+FILE=SD卡: 写16页
Disk-1001-1-0
AT+FILE=SD卡: 读1页数据正确
AT+FILE=SD卡: 读2页数据正确
AT+FILE=SD卡: 读3页数据正确
AT+FILE=SD卡: 读4页数据正确
AT+FILE=SD卡: 读5页数据正确
AT+FILE=SD卡: 读6页数据正确
AT+FILE=SD卡: 读7页数据正确
AT+FILE=SD卡: 读8页数据正确
AT+FILE=SD卡: 读9页数据正确
AT+FILE=SD卡: 读10页数据正确
AT+FILE=SD卡: 读11页数据正确
AT+FILE=SD卡: 读12页数据正确
AT+FILE=SD卡: 读13页数据正确
AT+FILE=SD卡: 读14页数据正确
AT+FILE=SD卡: 读15页数据正确
AT+FILE=SD卡: 读16页数据正确
```

17.7 打开 source/user_testapp.c 查看 User_FileAppTest 函数，找到 SD 卡文件读写测试，调用 APP_FileCtrl_Test 函数：实现 U 盘文件读写测试(当 APP_SD_FILE_TEST_EN 使能时，APP_SD_FILE_TEST_MODE 是 1)

17.8 打开 libapp/file_app.c 查看 APP_FileCtrl_Test 读写测试函数；

18. SDRAM 读写测试

18.1 测试功能：开机自动进行 SDRAM 读写测试。

18.2 打开 config/AMKN8639_Config.h 查看 SDRAM 配置定义, 不要修改

18.3 打开 config/test_config.h 查看 SDRAM 测试使能配置, 可以关闭和打开该测试功能。

18.4 开机后测试结果显示:



```
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 嘉立创PCB打样
AT+INFO=TIM12 Init, OK
AT+INFO=TIM13 Init, OK
AT+INFO=LPTIM5 Init, OK
AT+INFO=FCLK1 Init, OK
AT+INFO=PWM1 Init, OK
A
[09:59:31.270]收<-◆AT+USB=USBH Init OK
AT+INFO=NET Init, OK
AT+INFO=LTDC Init, OK
AT+INFO=DMA2D Init, OK
AT+FT5XX=Init OK!

[09:59:31.451]收<-◆AT+INFO=EEPROM从地址1200写入256字节数据
AT+INFO=EEPROM读写测试 OK!

[09:59:31.642]收<-◆AT+SDRAM=写入时间(32MB):184ms,速度:173MB/秒
[09:59:31.898]收<-◆AT+SDRAM=读取时间(32MB):256ms,速度:125MB/秒
AT+SDRAM=读写测试OK!
AT+WiFi Pals Init OK
USB Device Connected
AT+AI=8,5015,40,30,34,35,28,30,29

[09:59:32.000]收<-◆AT+FCLK1=F1,0.00,0.00,0.00,0.00
AT+FCLK1=F2,0.00,0.00,0.00,0.00,0.00
AT+FCLK1=F3,0.00,0.00,0.00,0.00,0.00

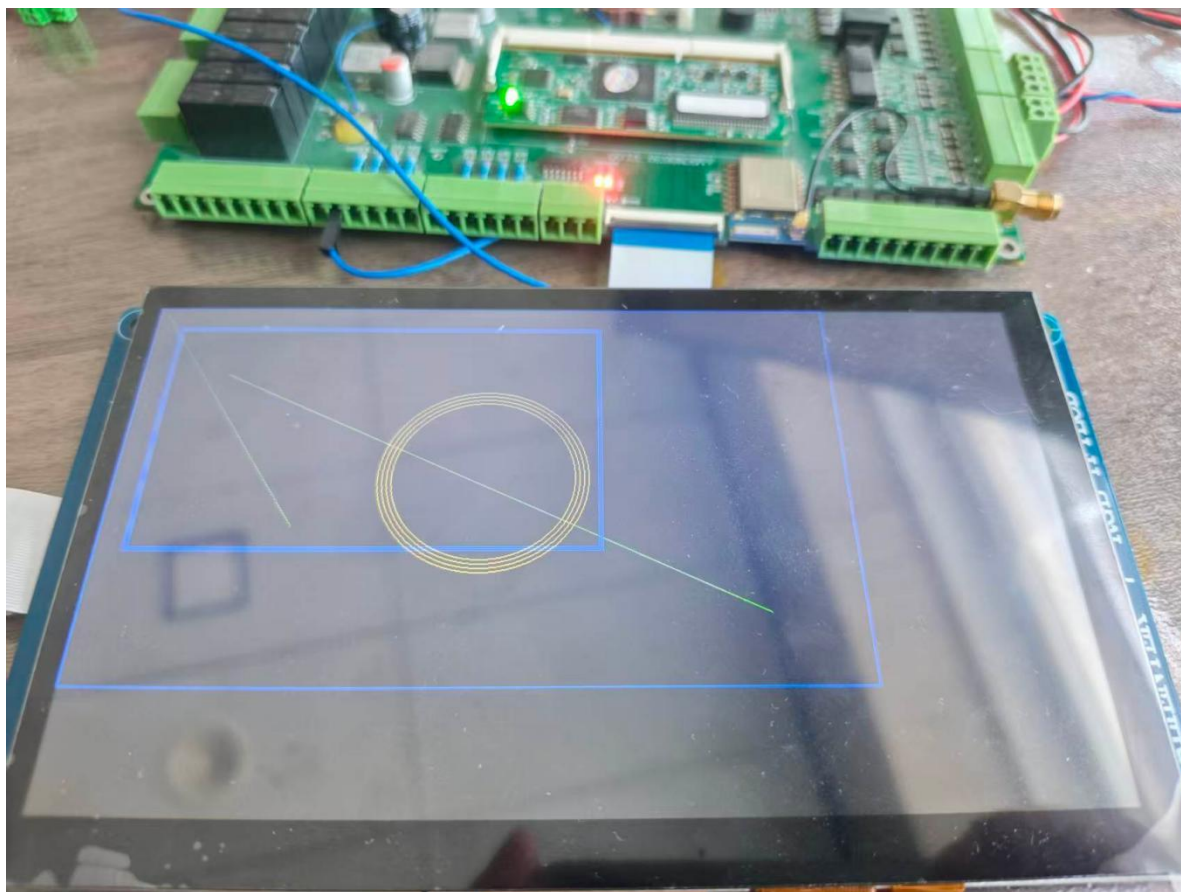
清除窗口 打开文件 发送文件 停止 清空发送区 最前 English 保存参数 扩展
端口号 COM19 USB-SERIAL CH340 HEX显示 保存数据 接收数据到文件 HEX发送 定时发送: 100 ms/次 加回车换行
```

18.5 打开 source/user_testapp.c 查看 User_AppTestInit 函数, 调用 SDRAM_AppTest 函数: 实现 SDRAM 读写测试(当 APP_SDRAM_TEST_EN 使能时)

18.6 打开 libapp/sdram_app.c 查看 SDRAM_AppTest 读写测试函数;

19. LCD 显示测试

- 19.1 测试功能：开机自动开启 LCD 显示测试。
- 19.2 打开 config/AMKN8639_IOConfig.h 查看 LCD 管脚定义(不可修改)
- 19.3 打开 config/AMKN8639_Config.h 查看 LTDC 配置, 要根据屏幕分辨率修改 LCD_PRODUCT 配置, 默认选择 1024*768 分辨率的 LCD;
- 19.4 打开\middlewares_app\lvgl_app\lvgl_config.h 将 LVGL_EN 设置为 0, 关闭 LVGL;
- 19.5 开机后 LCD 结果显示:



- 19.6 打开 source/TaskLCD.c 查看 App_TaskLCD 函数, 调用 User_LCDProcess 函数: 实现 LCD 显示;

附录：文档修改记录

序号	版本	时间	更新内容
1	V1.00	2026. 7. 24	正式发布。