BC7215A 离线空调码库

ESP-IDF(C++/FreeRTOS)演示程序

及

C++库

说明书

- 直接使用 idf 烧录/调试终端，可用于任何 ESP32
- 演示空调控制及红外指令解码等码库全部功能

前言

此演示程序基于 ESP-IDF V5.5.4 创建，直接使用烧录/调试终端为交互操作窗口，硬件连接上使用任意 4 个 I/O 口，由程序中定义，移植到不同 ESP32 型号时，只需使用 idf.py set-target <芯片型号> 改变芯片型号，以及改动 main.cpp 中相应行 I/O 口配置，即可完成移植。

例子中将标准的 C 语言 BC7215A 空调码库做了 c++包装，形成 BC7215 和 BC7215AC 两个类，BC7215 类为 BC7215 芯片的基础驱动类，可以单独使用，BC7215AC 为空调控制类，使用 BC7215 类为其成员，同时也简化了用户接口，同时代码也按照 ESP-IDF 的 FreeRTOS 环境做了适配。

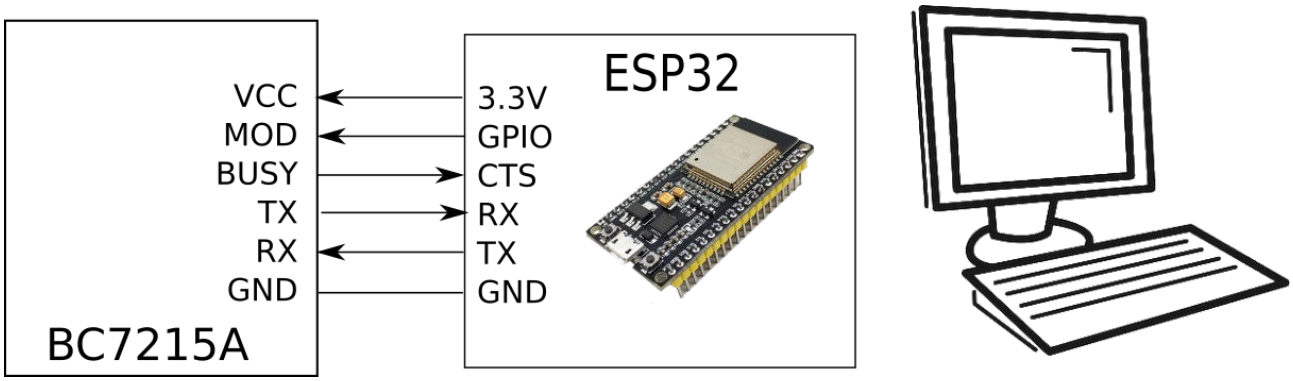
其他使用 c++和 FreeRTOS 的开发环境，也可参考本例子。

硬件连接与配置

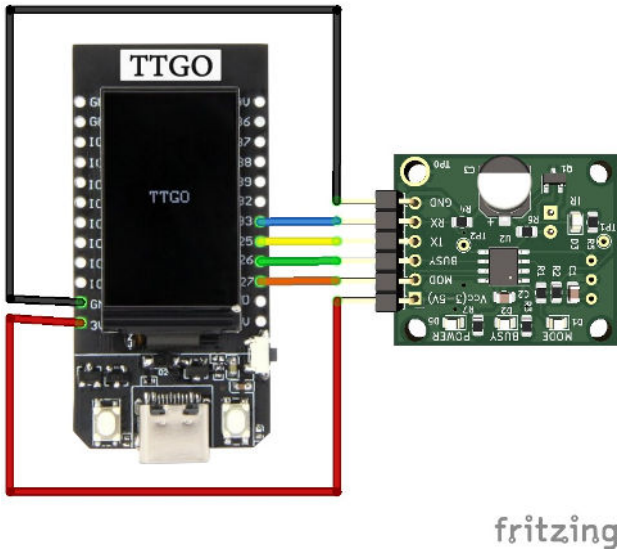
运行演示共需以下 I/O 资源:

一个带有 CTS 信号的 UART : 用于连接 BC7215A

GPIOx1 : 配置为输出，连接于 BC7215A 的 MOD



本例子中采用与 Arduino 例子中同样的 ESP32 TTGO T-Display 模块和连接，用户可以很方便地改用自己的模块和连接。



目录结构

```
Project_Name (项目根目录)
├── main
│   ├── include
│   │   └── bc7215_lib_config.h     <-- BC7215 驱动配置文件
│   │
│   ├── main.cpp                    <-- 主程序
│   └── CMakeList.txt
│
├── components
│   ├── bc7215
│   │   ├── include
│   │   │   └── bc7215.hpp
│   │   │
│   │   ├── bc7215.cpp              <-- BC7215 驱动 C++包装
│   │   └── CMakeLists.txt
│   │
│   ├── bc7215ac
│   │   ├── include
│   │   │   └── bc7215ac.hpp
│   │   │
│   │   ├── bc7215ac.cpp            <-- 空调码库 C++包装
│   │   └── CMakeLists.txt
│   │
│   └── bc7215_ac_lib
│       └── CMakeLists.txt
│
├── sdkconfig
└── CMakeList.txt

bc7215_ac_lib               <-- 外部目录，空调码库源文件
├── bc7215_types.h
├── bc7215_lib.h
├── bc7215_ac_lib.h
├── bc7215_lib.c
├── bc7215_ac_lib.c
└── (其他文件)
```

使用

编译下载

特别提醒：因使用了 FreeRTOS 和串口的硬件流控制，bc7215_lib_config.h 的内容和模板文件中有区别。

根据你的硬件连接修改 main.cpp 中的引脚定义后，和普通 ESP-IDF 项目一样，首先设置所使用的处理器：

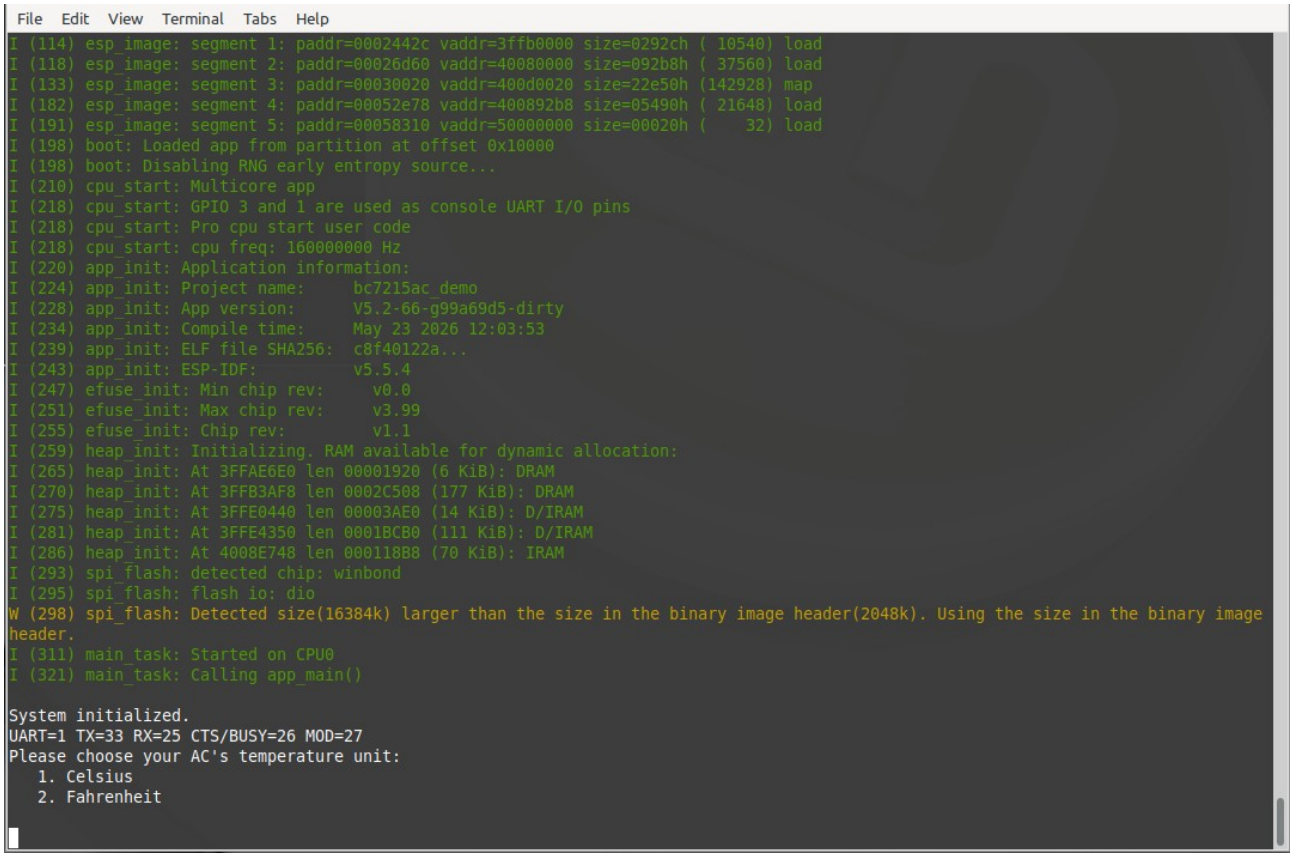
$> idf.py target <芯片型号>　<==芯片型号可以是 ESP32, ESP32-C3 等等任何 ESP32 系列

$>idf.py build　　<==编译项目

$>idf.py -p <串口设备> flash monitor　<==下载并进入调试终端

使用方法

例程运行后，会首先要求用户选择所控空调的温度制式是摄氏度还是华氏度，因为不同制式空调将需要调用不同的函数。

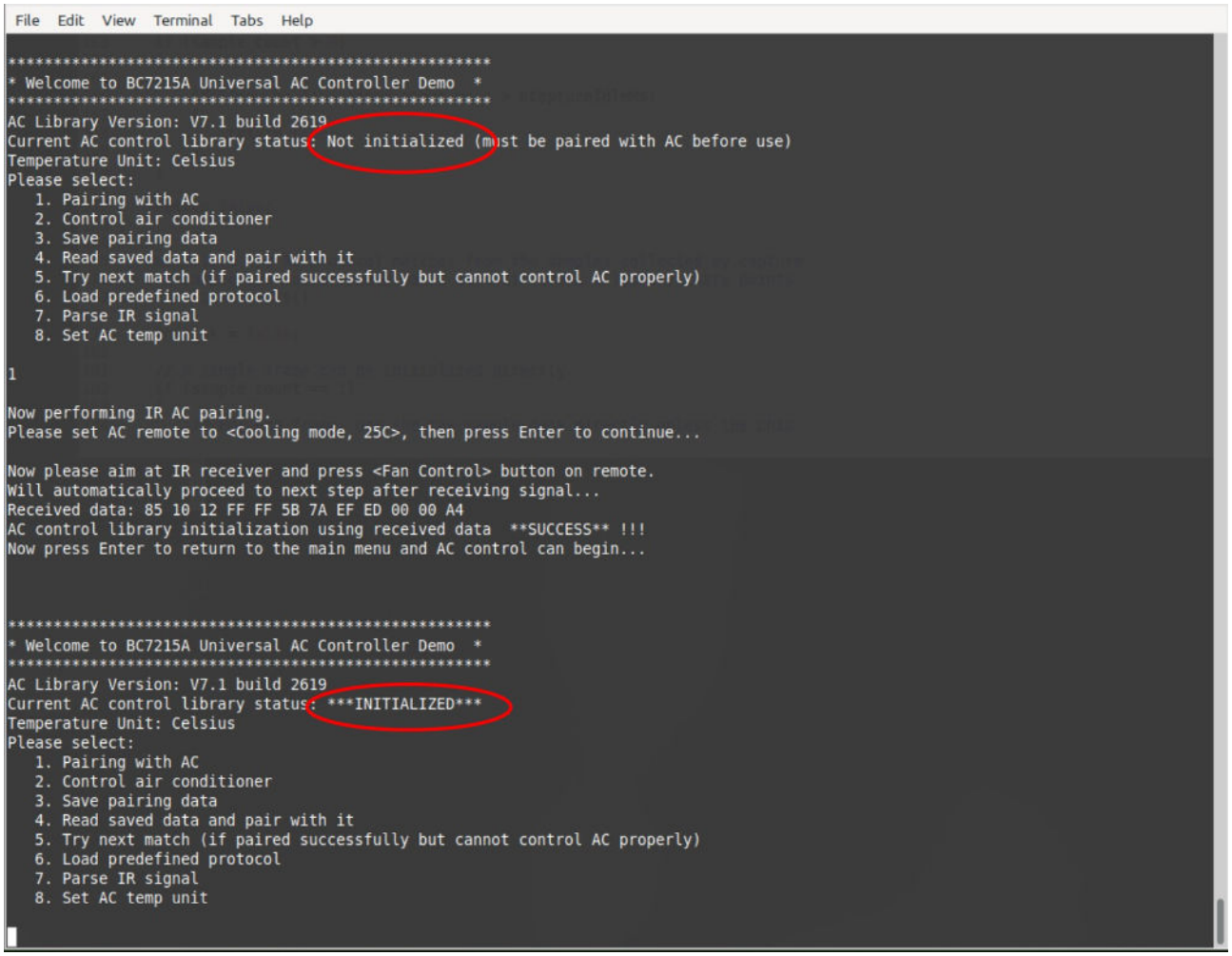


之后，在终端中显示主菜单。主菜单上会显示码库版本及当前的空调码库状态（是否已经初始化，即是否已与空调配对）。

显示主菜单后，在终端中，输入菜单编号，并回车，即可进入对应功能。

1. 初始化（配对）

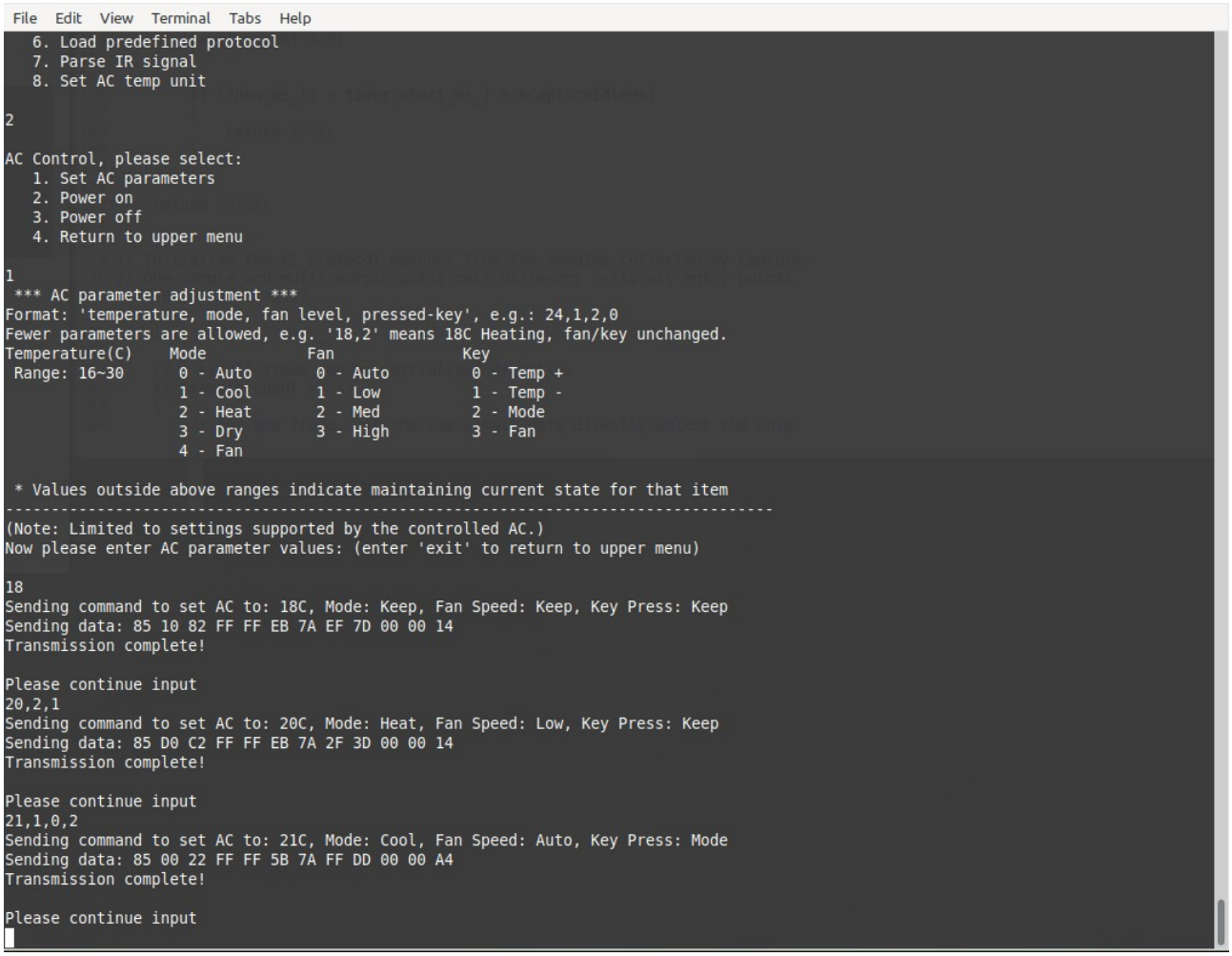
程序启动后，应首先与空调完成配对（初始化），才可操作其它功能。操作流程与 PC 上 DEMO 软件基本一致，用户可对照 BC7215A DEMO 板说明书中有关 DEMO 软件的部分作为参考。本例程中，红外信号采集和“初始化”合并成一个“配对”操作。



菜单选择操作后，均会有下一步操作的提示。上图显示了配对的过程。

2. 控制空调

完成初始化后，即可控制空调。主菜单选择‘2’进入空调控制子菜单



空调控制菜单有 4 项，选择‘1’进入空调参数控制页面，‘2,3’控制空调开关，‘4’返回主菜单。
在参数控制页面，使用方法为依次输入“温度，模式，风力，遥控器按键”4 个参数，中间用半角逗号隔开。支持部分输入，如仅需改变温度时，可以仅输入温度，这种情况下其它设置将保持不变。

输入英文“exit"，则退出空调参数控制，返回上层菜单。

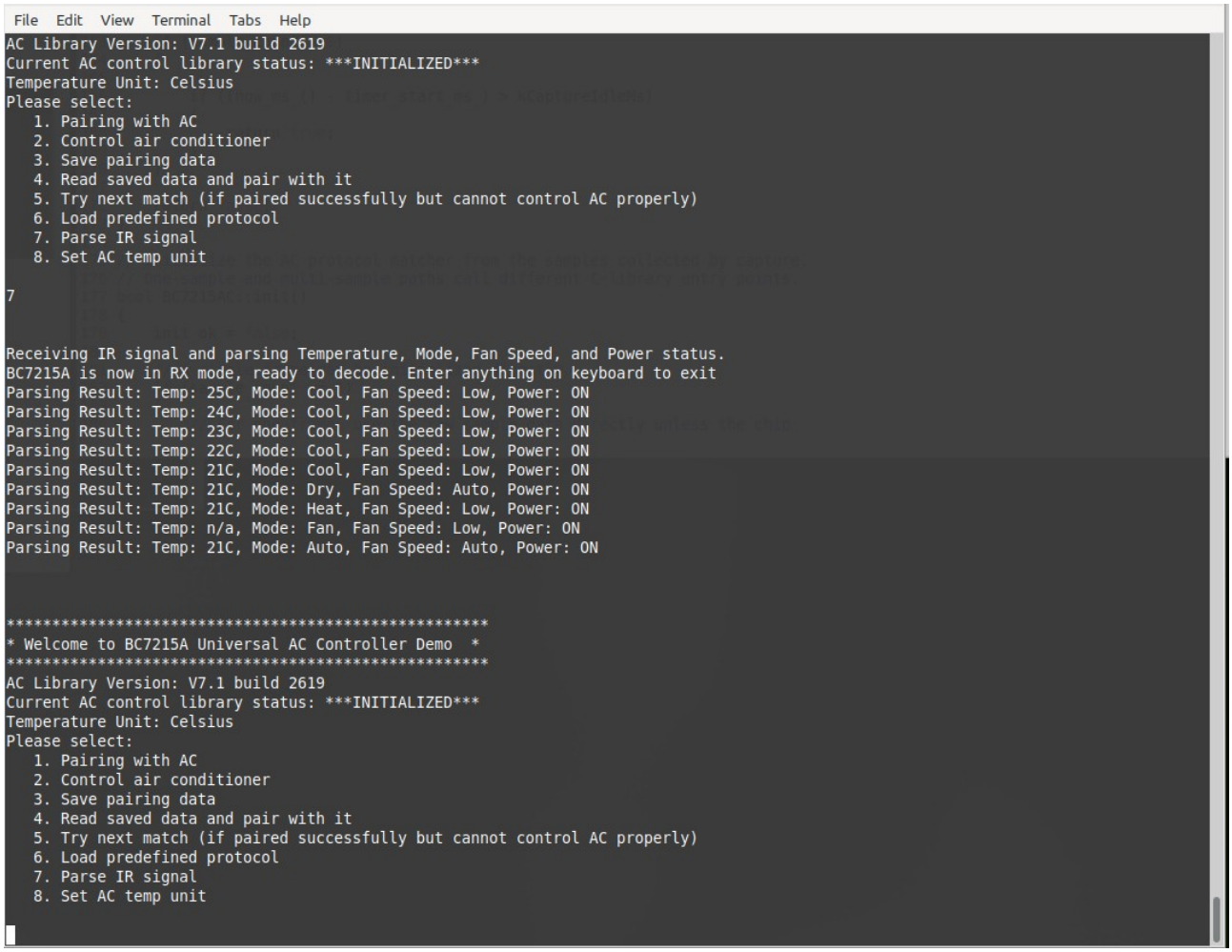
输入的合法数值范围在进入参数控制页面时有显示，超出范围的输入被认为是该项设置保持不变。输入设置值后，终端会显示所接收到设置内容，并立刻进入红外发射状态，并且显示所发射的数据。发射结束后，会进入等待下一个输入的状态。

使用界面参见下图:

3.红外指令解析

配对成功后，还可进行红外指令的解析。注意，解析仅对所配对的空调遥控器有效。主菜单选择
7-Parse IR Signal
即进入解析模式，进入解析模式后，BC7215A 转为接收状态，此时如果对着红外接收头按遥控器按键，程序即可将指令中的“温度、模式、风力、电源状态”4 个参数解析并打印出来。

解析状态按回车键，即退回主菜单。



4.其它

存储和读取配对信息，使用了 ESP-IDF 框架的 NVS, 但存储功能不在空调库范畴内，用户也可使用其他的方法存储配对数据。

有关查找下一匹配和加载预定义协议操作的作用，请参考《BC7215A 离线空调红外码库说明书》。

C++类的使用

C 语言的库被包装为了 C++的类，均放在"bc7215"的 namespace 内。

bc7215::BC7215 类

根据 ESP-IDF 的环境，整合了串口的操作部分，用户无需再提供底层串口函数。构造函数参数为串口号和引脚分配。

BC7215 类，提供了和 C 驱动相同的函数，提供的公共函数与 C 语言版函数对应，功能也相同。不过提供了以下函数的重载，可以直接输入 bc7215DataMaxPkt_t 和 bc7215FormatPkt_t 类型数据的引用，免去指针类型转换的繁琐：

uint8_t get_data(bc7215DataMaxPkt_t& target);

uint8_t get_format(bc7215FormatPkt_t& target);

void ir_tx(const bc7215DataMaxPkt_t& source);

bc7215::BC7215AC 类

包装了 C 语言的空调码库，但提供了更易用的功能，比如：采样-配对的操作，简化为了 start_capture() 和 init()两个函数的调用，具体执行逻辑在类内部完成。BC7215 类是其中的一个成员，使用中用户无需再单独定义 BC7215 类变量。

公共成员变量：

BC7215AC 类提供了几个可从外部访问的变量，以方便用户使用：

bool init_ok; // 是否已经初始化（配对）

uint8_t sample_count; // 采样到的数据段数，最多 4 段

uint8_t sample_status[4]; // 每段的 status

bc7215DataMaxPkt_t sample_data[4]; // 每段的数据包

bc7215FormatPkt_t sample_format[4]; // 每段的格式包

采样结束后，如用户需要显示所接收到的数据的内容等，可以直接访问这几个变量。

公共成员函数：

构造函数：参数为串口号和引脚分配，这些参数会传递给内部的 BC7215 类成员，完成串口设置。

初始化函数：

esp_err begin() ;

使用前，必须先呼叫 begin()函数，完成必要的初始化。如果不成功，会返回错误代码。

温度制式有关函数：

void set_fahrenheit();

void set_celsius();

bool is_celsius();

空调温度制式设置和查询函数，如果改变了制式，会令配对失效，需要重新配对。

采样相关函数：

void start_capture(uint8_t rx_mode);

void stop_capture();

bool signal_captured();

红外信号采集相关指令，开始采集指令的参数为 BC7215 芯片进入 RX 状态后的模式。进行配对时，必须使用复合模式(rx_mode=1), 进行红外解析时，可用简单模式，减少数据传输。

对多段数据的处理，已经包含在函数中，用户不必再关心多段数据采样的处理流程。

初始化(配对)相关函数：

bool init();

bool init(const bc725DataMaxPkt_t& data, const bc7215FormatPkt_t& format);

bool match_next();

bool init_predefined(uint8_t index);

初始化（配对）相关函数，操作流程已经包装在类内，配对过程简化为：start_capture(1) ==> 检查 signal_captured() ==> stop_capture() ==> init() 4 个函数呼叫。

对于使用保存的配对信息进行上电初始化的场景，使用带参数的

init(const bc725DataMaxPkt_t& data, const bc7215FormatPkt_t& format);

欲使用预定义协议数据初始化时，init_predefined(uint8_t index) 一个函数即可完成，参数为数据的序号。

match_next() 的作用和 C 语言版的 bc7215_find_next_match()一样。

空调控制相关函数：

const bc7215DataVarPkt_t* set_to(int temp, int mode = -1, int fan = -1, int key = 2);

const bc7215DataVarPkt_t* on();

const bc7215DataVarPkt_t* off();

封装后的 C++类，已经将 C 语言码库的数据产生和红外数据的发射整合在一起，用户不必再关心如何将得到的指令数据处理并用红外发射出去。所有空调的控制指令浓缩为这三个函数，开关机和设置状态，呼叫这三个函数后，会自动完成数据的发送。

类内部会暂存空调最后的状态，关机后再开机时，会恢复之前的设置。处于解析状态时，每次成功解析出的空调设置，也会被同步给内部暂存的空调状态。

* 多数空调并无专用的开机指令，关机状态下发送设置温度等指令就会自动开机。

红外解析函数：

bool parse(int& temp, int& mode, int& fan, int& power);

解析函数基本同于 C 语言版，但输出的温度根据空调的温度制式做了还原，即摄氏时 temp 的范围为 16-30，华氏时输出 temp 的范围为 60-88.

其它函数：

uint8_t predefined_count() const; // 获取预定义协议数量

const char* predefined_name(uint8_t index) const; // 获取预定义协议名称

bool is_busy() const; // 查询是否忙碌，可用于查询红外指令是否发送完成

```
const bc7215DataVarPkt_t* data_packet() const;          // 获取基础数据包
const bc7215FormatPkt_t*  format_packet() const;        // 获取基础格式包
const char*               lib_version() const;  // 查询码库版本
const BC7215& driver() const;     // 获取底层驱动，当需要直接操作底层驱动库时，可用此函数
```

需要保存初始化信息供下次上电使用时，可通过获取基础数据包和基础格式包指令，不建议采样后直接从公共变量中读取，因为有时会有多个数据包。另外初始化成功后，在保存之前，不应该执行解析功能，因为解析功能会替换掉基础数据包。