

BC7215(A)

驱动库

V4.x

使用说明书

目录

简介.....	3
使用.....	3
配置文件.....	3
数据类型.....	5
用户需提供的硬件接口函数.....	6
I/O 口操作函数.....	6
UART 操作函数.....	6
用户接口函数.....	7
1. 串口数据处理函数.....	7
2. 状态控制函数.....	7
3. 查询函数.....	8
4. 接收相关函数.....	9
5. 发射相关函数.....	10
6. 工具函数.....	11
7. 硬件接口配置函数.....	12

简介

本 BC7215(A) C 语言驱动库提供了和 BC7215(A)的接口，用户使用 BC7215(A)时，调用本驱动库的函数，即可完成面向 BC7215(A)芯片的各种操作。

本驱动库包含应用 BC7215(A)所需的各种操作，但不包括涉及硬件的操作，所有和硬件的接口，需要用户通过函数方式提供。

使用

本库以 C 语言源文件的形式提供，共 4 个文件：

bc7215_lib_config.h

bc7215_types.h

bc7215_lib.h

bc7215_lib.c

使用时，只需将 bc7215_lib 目录拷贝到用户的项目源文件目录中，根据根据不同的开发环境的要求，用户可能需要将 bc7215_lib.c 文件加入项目的源文件清单，并将 bc7215_lib 目录加入到用户项目的 include 文件搜索目录。

用户只需 c 源文件中添加

```
#include "bc7215_lib.h"
```

即可在程序中使用本库所提供的各个函数。

本驱动库要求编译器至少支持 C89 或以上的 C 语言版本。

配置文件

本驱动库包含一个配置文件 bc7215_lib_config.h，其中定义了一些本驱动库有关的参数，用户使用前可根据需要做调整。主要配置参数如下：

TX_HW_FLOW_CONTROL

这个设置主要涉及向 BC7215(A)芯片的**数据发送**，即系统的 UART 发送是否使用了硬件流控制。如果硬件系统支持串口流控制，比如在电脑上以及很多单片机如 STM32 系列，则建议使用硬件流控制来控制向 BC7215(A)的数据发送，即将 BC7215 的 BUSY 信号作为串口发送的许可信号（在电脑上通常为 CTS 信号）。使用硬件流控制可以简化软件设计，降低系统开销。此值定义为 1，表示使用了硬件流控制，默认为 0。

此处的硬件流控是指对本驱动库而言，即不用 BC7215(A)驱动库去关心流控制，而由更底层的系统完成，如果用户通过软件实现了由 BUSY 信号控制的数据发送，能够保证每个字节发送到串口数据线上时，BUSY 都为低电平，则对驱动库而言，也等同于有硬件流控，可选择此项。

当不使用硬件流控制时，用户须提供一个读取 BC7215(A) BUSY 引脚状态的函数。

USE_UART_INTERRUPT

此设置主要涉及从 BC7215(A)的**数据接收**，即是否使用中断来处理串口数据。在单片机系统中，多使用中断来处理接收到的串口数据，而在有操作系统的复杂系统如 Windows 和 Linux，包括 Arduino 系统中，系统会提供底层的串口驱动，接收的数据自动写入串口缓存，用户程序只需读取串口缓冲区，而无需直接操作串口硬件。此值定义为 1，表示使用中断方式来接收串口数据，默认为 1。

此值定义为 1 时，用户应当在串口接收中断中呼叫 `bc7215_process_uart_data()` 函数，且用户须提供两个使能和禁止串口中断的函数，驱动库会在进行特定操作时禁止串口中断，以防止冲突。

如果所用系统具有自动将串口接收数据存入缓冲区的功能（包括用户自己通过软件实现了这样的功能），串口数据使用者只需定期或者由某种事件驱动去读取串口缓冲区，则此项选 0。

PROCESS_IN_DIFFERENT_THREAD

此设置主要适用与使用实时多任务操作系统的情况，即是否串口数据处理（即呼叫 `bc7215_process_uart_data()` 函数）在不同的线程完成。在支持多线程的系统中，即便不使用中断，串口数据处理和 BC7215(A)的主要操作，仍然可能会被放在不同的线程（不建议），这种情况下，和使用中断系统类似，用户必须提供两个互斥锁的操作函数，在驱动库初始化时，通过 `bc7215_config_mutex_lock_funct()` 和 `bc7215_config_mutex_unlock_funct()` 两个函数通知到驱动库，驱动库在相关操作时会呼叫这两个函数，以确保不会产生资源访问上的冲突。此值为 1 时，表示 `bc7215_process_uart_data()` 呼叫和 BC7215 其他操作在不同的线程，默认值为 0。

ENABLE_RECEIVING

这个参数控制是否使能驱动库的接收解码处理功能，包括接收原始数据包，接收格式信息包等。这个参数默认为 ‘1’，即使能状态，如果用户不需要 BC7215 的接收解码功能，可以将其修改为 ‘0’。禁止接收功能后，相关的函数也将变得不可用。因为接收解码功能占用了大多数所需的内存和程序内容，因此禁止接收功能，将大大缩小本驱动库的体积和所占用的内存。

ENABLE_FORMAT

在接收功能下，还有个进一步的控制参数，控制是否使能接收格式信息。默认值为 ‘1’，为使能状态，如果不需要，则可以改为 ‘0’。有些应用，如应用于数据通讯时，并不需要获取红外信号的格式信息，这时就可以将此功能关闭，因为格式信息包为 33 个字节，因此将节省 33 个字节的 RAM 和一部分程序空间。

ENABLE_TRANSMITTING

这个参数控制是否使能红外发射相关功能，默认为 ‘1’ 使能状态，改为 ‘0’ 可以关闭。关闭发射相关功能可以节省一部分程序空间。

BC7215_MAX_RX_DATA_SIZE

用字节数表示的可接收的最大原始数据包的长度，范围为 1-512, 实际遥控器所发出的数据长度，影音设备一般在 8 字节以内，空调类的，一般在 48 字节以内。此定义值越大，本驱动库将占用更多的内存。

BC7215_CRC8_POLY

本函数库提供了一个 CRC-8 计算的函数，此参数定义了 CRC-8 计算的多项式值，默认值为 0x07, 用户可根据自己需要修改。

数据类型

本驱动库定义了两种数据类型：

原始数据包，原始数据包所占用的内存大小由配置文件中的用户定义常量所决定：

```
typedef struct bc7215DataMaxPkt_t
{
    uint16_t bitLen;
    uint8_t data[BC7215_MAX_RX_DATA_SIZE];
} bc7215DataMaxPkt_t;
```

此类型定义了能处理的最大数据包。

```
typedef struct bc7215DataVarPkt_t
{
    uint16_t bitLen;
    uint8_t data[1];
} bc7215DataVarPkt_t;
```

此类型定义了数据包的结构，用于处理可变长度的数据包，主要以指针的方式使用。

格式数据包，格式数据包为固定大小，占 33 个字节：

```
typedef struct {
    union {
        struct {
            uint8_t sig : 6;
            uint8_t c56k : 1;
            uint8_t noCA : 1;
        } bits;
        uint8_t inByte;
    } signature;
    uint8_t format[32];
} bc7215FormatPkt_t;
```

同时，驱动库还定义了 3 种函数指针类型，用于表示用户所提供的硬件接口函数的签名特征，分别为无输入参数，无返回值；1 个输入参数（8 位宽度），无返回值；以及无输入参数，返回 8 位数值三种类型：

```
typedef void      (*voidFunction_t)      (void);  
typedef void      (*uint8ParamFunction_t) (uint8_t);  
typedef uint8_t   (*readIOFunction_t)     (void);
```

用户需提供的硬件接口函数

用户所提供的硬件相关函数，包括以下两类，详情请参见《[硬件接口配置函数](#)》一节：

I/O 口操作函数

包括 I/O 口的写入和状态读取，这类操作通常仅需一条指令即可完成，但用户需将其封装为函数，以便驱动库调用，包括：

MOD 引脚置高函数 —— 功能为将 BC7215(A)的 MOD 引脚置为高电平，如果 MOD 引脚为通过硬件接地或拉高，则不必提供，但这种情况下请注意不可调用 `bc7215_set_rx()` 和 `bc7215_set_tx()` 函数，否则将导致程序崩溃。

MOD 引脚置低函数 —— 功能为将 BC7215(A)的 MOD 引脚置为低电平，如果 MOD 引脚为通过硬件接地或拉高，则不必提供，但这种情况下请注意不可调用 `bc7215_set_rx()` 和 `bc7215_set_tx()` 函数，否则将导致程序崩溃。。

MOD 引脚读取函数 —— 功能为读取当前 MOD 引脚的状态，**此函数必须提供**，如果 MOD 引脚硬件上为固定高或低，函数直接返回 0 或 1 即可，不必读取 MOD 引脚。

BUSY 引脚读取函数 —— 功能为读取当前 BUSY 引脚的状态，**当配置为不使用硬件流控制时，必须提供此函数**，此时所有涉及到向 BC7215(A)发送数据，包括接收模式下改变工作模式函数 `bc7215_set_rx_mode()`，均需要此函数，如果未连接 BUSY 引脚，则该函数应设置为直接返回 '0'。

UART 操作函数

BC7215(A)在接收模式时，会主动通过 UART 发送所接收到的数据和红外格式信息，因此要求主机一侧能随时接收 UART 的输入数据。用户所提供的函数包括：

以下 3 个函数仅有当配置为“使用串口中断”时，才必须提供：

UART 中断使能函数 —— 功能为使能 UART 中断，如果所用硬件 UART 包含多种不同的中断，此处特指 UART 接收中断，如果不使用红外接收解码功能，则不必提供。

UART 禁止中断函数 —— 功能为禁止 UART 中断，如果不使用红外接收解码功能，则不必提供。

UART 接收中断处理函数 —— 此函数不用提供给本驱动库，但用户必须在中断处理函数中调用本驱动库的“串口数据处理函数”。

下面函数为必须提供：

UART 单字节发送函数 —— 用户必须提供一个串口发送函数，功能为通过 UART 向 BC7215(A)发送一个字节。在不同的系统中，对此函数的要求有所不同：

- 在串口发送不使用硬件流控制的系统中，此函数要求必须待该字节完全从串口发出后，才从函数返回。
- 在使用硬件流控制的系统中，此函数仅需将数据写入发送缓冲区即可返回，不必等待数据实际发送完成。

所有涉及到向 BC7215(A)发送数据，包括接收模式下改变工作模式函数 `bc7215_set_rx_mode()`，均需要此函数。

用户接口函数

用户接口函数为驱动库向用户提供的函数，用户在程序中可直接调用，操作 BC7215(A)完成各种功能，包括以下 7 类：

1. 串口数据处理函数
2. 状态控制函数
3. 查询函数
4. 接收相关函数
5. 发射相关函数
6. 工具函数
7. 硬件接口设置函数

1. 串口数据处理函数

```
void bc7215_process_uart_data(uint8_t data);
```

在使用串口中断的系统中，用户需要在串口接收中断中，调用此函数，输入参数为每个收到的串口数据。

在由系统管理串口底层操作，用户不直接访问串口硬件的系统中，用户需要在读取串口输入缓冲区时，依次用收到的每个字节作为参数，呼叫此函数，直至接收缓冲区清空。

2. 状态控制函数

用户通过此类函数控制 BC7215 的工作状态。

```
void bc7215_set_rx(void);
```

设置 BC7215(A)的工作模式为接收(红外解码)模式。如果调用此函数时 BC7215(A)正在进行红外发射，则此函数被调用后，BC7215(A)可能需要完成正在发送的红外比特才能切换到接收状态，最长可能需要 20ms 的时间，用户应保证此期间不对 BC7215(A)进行其他的操作。。

```
void bc7215_set_tx(void);
```

设置 BC7215(A)的工作模式为发射(红外编码)模式。此函数调用后，BC7215(A)芯片可能需要最多 70ms 的时间完成模式切换，用户应保证此期间不对 BC7215(A)进行其他的操作。

```
void bc7215_set_shutdown(void);
```

在发射模式下，设置 BC7215(A) 进入关机模式。调用此函数后，向 BC7215(A) 发送 F7 00 指令。此函数只在发射模式下起作用，如果在接收模式下调用，因为最后发送的数据为 0x00，BC7215(A) 的接收模式将变为简单模式(详情请参阅 BC7215(A) 数据手册)。此函数调用后，用户可以查询指令执行状态。

```
void bc7215_set_rx_mode(uint8_t mode);
```

接收模式下，设置接收解码模式(详情请参阅 BC7215(A) 数据手册)，mode 的最低两位决定指令执行后的工作模式。

3. 查询函数

用户通过此类函数查询 BC7215(A) 的工作状态。

```
uint8_t bc7215_data_ready(void);
```

查询是否从 BC7215(A) 接收到了有效的红外数据包。当有有效数据时，返回 1，否则返回 0。此查询仅在接收模式有效，如果在发送模式调用此函数，将永远返回 0。如果配置中禁止了接收功能，则此函数不可用。

有三种情况下红外数据有效的状态会被清除：

1. 串口接收到了新的数据
2. 调用了获取数据包函数 `bc7215_get_data()`
3. 调用了清除数据包函数 `bc7215_clr_data()`

```
uint8_t bc7215_format_ready(void);
```

查询是否从 BC7215(A) 收到了格式数据，当为复合模式时，BC7215(A) 除了输出原始数据外，还输出所收到红外信号的格式信息。一般来说如果收到格式信息，之前必定也收到原始数据，但有一种情况是原始数据加格式信息的总长度超过了驱动库缓冲区的长度，这种情况下，之前收到的原始数据会被覆盖，因此只有格式信息是可用的。这种情况下要想获取原始数据，需要将 BC7215(A) 设置为简单模式，重新接收红外信号。

当有有效格式数据时，返回 1，否则返回 0。此查询仅在接收模式有效，如果在发送模式调用此函数，将永远返回 0。如果配置中禁止了接收功能，则此函数不可用。

有三种情况下格式数据有效的状态会被清除：

1. 串口接收到了新的数据
2. 调用了获取格式包函数 `bc7215_get_format()`
3. 调用了清除数据包函数 `bc7215_clr_format()`

```
uint8_t bc7215_cmd_completed(void);
```

发射状态下，查询指令是否已经执行完成。可以查询的是发射指令和关机指令。

BC7215(A) 芯片内部具有 16 字节的接收缓冲区，尽管红外发送的速率比较低，但数据量在 16 个字节以内时，发送函数会几乎立刻返回，但实际数据通过红外线发送过程，由 BC7215(A) 芯片完成，则需要较长时间。用户有时需要知道发送完成的时间，比如需要保证连续两次发射的时间间隔时，则可用此函数查询上一次发射的完成时间。

关机指令会立即执行，用户可通过此函数查询 BC7215(A) 是否已经进入关机状态。

返回值为 1 时，表示指令执行完成，为 0 时，表示尚未完成。在接收状态执行此函数，将始终返回 1。


```
uint8_t bc7215_is_busy(void);
```

此函数为 V4.0 版新增，在发射模式时，其功能等同于 `bc7215_cmd_completed()`，但返回为反相逻辑的结果。在接收模式时，此函数返回值代表 BC7215(A) 芯片是否处在数据包或格式包的接收过程中，即已经收到数据，但尚未收到结束标志 0x7A 的情况。返回值为 0 时，表示芯片空闲，为 1 时表示芯片忙。

4. 接收相关函数

```
uint16_t bc7215_get_len(void);
```

获取所接收到的原始数据包的数据长度，此长度为实际数据的比特数，即 `bc7215DataVarPkt_t` 中的 `bitLen`，**`bc7215_get_data()`** 函数实际拷贝的数据长度，为这个值所对应的字节数+2。如这里 `bitLen` 为 9 时，原始数据需要 2 个字节，实际拷贝的字节数将为 4。

如果原始数据包不可用，返回结果将为 0。

用户可利用这个函数在获取原始数据前检查数据长度，防止内存溢出，或动态分配内存。

如果配置中禁止了接收功能，则此函数不可用。

```
uint16_t bc7215_dpkt_size(void);
```

获取所接收数据包的大小，以字节为单位。此函数与上面获取比特长度函数等价，但得到的是整个数据包的字节数，免去用户自行转换的步骤。因为所接收的数据包由红外发射端决定，有可能会收到超出预期的数据，用户可以在收取数据包前检查数据包大小，防止内存溢出。

如果配置中禁止了接收功能，则此函数不可用。

```
uint8_t bc7215_get_data(bc7215DataVarPkt_t* target);
```

读取原始数据包函数。输入参数为一 `bc7215DataVarPkt_t` 类型的指针，指令执行后，所接收到的原始数据包会拷入该指针指向的目的地址。返回值为原始数据包的状态特征字，用以快速判断所接收数据的状态。如果缓存中的原始数据已经不可用（如已被新数据覆盖等情况），会返回 0xff。

此函数执行后，将清除 `bc7215_data_ready()` 的状态。

用户必须保证目标 `target` 所指向的内存空间大小能够容纳所接收到的原始数据，如果所接收到的数据长度大于 `target` 所指向的内存空间，执行此函数会造成内存溢出，产生不可预期的后果。用户应先用 `bc7215_get_len()` 函数获取数据长度，申请空间或检查空间大小后再调用此函数。

如果配置中禁止了接收功能，则此函数不可用。

```
uint8_t bc7215_get_raw(void* addr, uint8_t size);
```

```
uint16_t bc7215_get_raw(void* addr, uint16_t size);
```

获取所接收数据包中的原始数据函数。此函数和上面读取数据包函数类似，不过此函数只读出原始数据而不包含位长度信息，而且可以输出到任意地址，更适合作为红外通信时使用。根据配置文件中的 `BC7215_MAX_RX_DATA_SIZE` 参数，当此值大于 255 时，输入 `size` 和返回值都是 8 位 `uint8_t` 类型，当大于等于 256 时，两个参数的类型变为 `uint16_t` 类型。

size 为希望读取的字节数，该值不一定与接收到的数据相同，可以小于或者大于实际接收到的字节数，小于时，按 size 的值返回字节数，当 size 大于实际接收到字节数时，仅读出实际接收到的字节数。函数的返回值，为实际读出的字节数。

如果配置中禁止了接收功能，则此函数不可用。

```
uint8_t bc7215_get_format(bc7215FormatPkt_t* target);
```

读取格式数据包函数。输入参数为一 bc7215FormatPkt_t 类型的指针，指令执行后，所接收到的格式数据包将会拷入该指针所指向的目的地址。返回值为格式数据包的特征字，如果缓存中的原始数据已经不可用（如已被新数据覆盖等情况），会返回 0xff。

此函数执行后，将清除 bc7215_format_ready() 的状态。

此函数固定向目标地址拷贝 33 个字节的数据，用户须保证目标地址有足够空间容纳格式数据包。

如果配置中禁止了接收功能，则此函数不可用。

5. 发射相关函数

```
void bc7215_load_format(const bc7215FormatPkt_t* source);
```

加载格式数据到 BC7215(A) 芯片。调用此函数后，会将指针 source 所指向的格式信息数据加载到 BC7215(A) 芯片。

如果配置中禁止了发射功能，则此函数不可用。

```
void bc7215_IR_tx(const bc7215DataVarPkt_t* source);
```

发射红外数据。调用此函数，会令 BC7215(A) 通过红外线发送 source 所指向的原始数据包。

发送所使用的格式，为最后一次加载的格式，如果是从接收模式切换到发射模式，且没有加载过格式数据，则会使用最后一次接收到的红外信号的格式。如果待发送数据少于 16 字节（128 比特），此函数会将数据写入 BC7215(A) 片内的缓冲区后立即返回，如果多于 16 字节，则在最后 16 个字节被写入缓冲区后返回。

具体红外发射所需的时间，由所使用的红外调制格式决定，通常每个字节需几个 ms。用户可以通过 **bc7215_cmd_is_completed()** 函数查询红外发射是否完成。

如果配置中禁止了发射功能，则此函数不可用。

```
void bc7215_send_raw(const void* rawData, uint16_t size);
```

发送原始数据。输入为指向原始数据的指针和数据长度，指针指向的数据不必为 BC7215(A) 的数据包格式。此函数便于用于数据通讯，但仅能按完整字节发送。数据尺寸最大为 512 字节，但因为红外发送速率不高，发送长数据包将会需要较长时间且容易受到外界干扰，建议每次发射控制在几十字节范围内。

```
void bc7215_set_C56K(bc7215FormatPkt_t* target);
```

设置格式数据包特征字的控制位，设置后其中 C56K 位将置位，加载数据包后，BC7215(A) 将使用 56K 载波发射。

```
void bc7215_clr_C56K(bc7215FormatPkt_t* target);
```

清除格式数据包特征字的控制位，设置后其中 C56K 位将被清除 (恢复缺省状态)，加载数据包后，BC7215 将使用 38K 载波发射。

```
void bc7215_set_NOCA(bc7215FormatPkt_t* target);
```

设置格式数据包特征字的控制位，设置后其中 NOCA 位将置位，加载数据包后，BC7215(A)输出将不使用载波，为纯高低电平输出。

```
void bc7215_clr_NOCA(bc7215FormatPkt_t* target);
```

设置格式数据包特征字的控制位，设置后其中 NOCA 位将被清除 (恢复缺省状态)，加载数据包后，BC7215(A)将使用 38k 或 56k 载波输出。

6. 工具函数

驱动库同时提供了一些工具函数，这些函数不直接操作 BC7215(A)芯片，而是用户在使用 BC7215(A)时经常会用到的一些操作。

当最大 BC7215_MAX_RX_DATA_SIZE 小于 256 字节时：

```
uint8_t bc7215_crc8(uint8_t* data, uint8_t len);
```

或者当最大数据包大于等于 256 字节时：

```
uint8_t bc7215_crc8(uint8_t* data, uint16_t len);
```

CRC8 计算函数。如果将 BC7215 应用于数据通讯，可能会需要给数据包加上 CRC 校验以提高可靠性，因为 BC7215 适宜使用的数据包一般较小，建议的数据长度在 16 个字节以内，而且因为红外通讯速率较低，因此使用 8 位 CRC 校验更符合需要，既能起到防止错误作用，又不会增加过多通讯时间。

函数有两个输入参数，第一个为指向数据的指针，第二个为需计算 CRC 的数据长度。这里的长度是指字节长度，而不是数据包中的位长度，本函数只能按字节计算 CRC。返回值为计算的 CRC 结果。CRC 计算的多项式，本驱动库默认为 0x07，在 bc7215_lib_config.h 文件中定义，用户可根据需要修改。

```
uint16_t bc7215_cal_size(const bc7215DataVarPkt_t* dataPkt);
```

计算数据包大小。这个函数返回 dataPkt 所指向的内存中的数据包的大小，包括原始数据部分和位长度部分，以字节为单位。

```
void bc7215_copy_dpkt(bc7215DataVarPkt_t* target, const  
bc7215DataVarPkt_t* source);
```

数据包拷贝函数。此函数将 source 所指向的数据包拷贝到 target 的地址。需要特别说明的是此函数支持 source 和 target 两个数据包重叠的情况，这种情况下实际上是完成了数据包的移动，比如 target 的地址可以仅比 source 相差 1 个字节，相当于将 source 数据包在内存里向前或者向后移动一个字节。

```
uint8_t bc7215_compare_dpkt(uint8_t sig, const bc7215DataVarPkt_t*  
pkt1, const bc7215DataVarPkt_t* pkt2);
```

比较两个数据包。此函数比较两个数据包是否相同。如果相同，函数返回‘1’，如果不同，返回‘0’。

红外数据的结尾可能不是完整的字节，此函数支持比较不完整的字节，因为不同的编码格式，数据有可能是 MSB 在前或者 LSB 在前，因此还需要特征字信息以确定数据的排列方向。函数假设两个数据包的特征字是相同的，如果两个数据包调制方式不同，其数据长度和数据很大可能也是不同的。

7. 硬件接口配置函数

此类函数告知驱动库用户提供的硬件接口函数，从而使得驱动库可以进行所需要的硬件操作。这些函数必须在使用所有其它库函数之前被调用。

```
void bc7215_config_uart_int_en_func(voidFunction_t);
```

配置 UART 中断使能函数。当使用串口中断时，用户必须提供一个功能为使能串口中断的函数。不同的控制器可能有不同的中断设计，如果 UART 有多种中断可供使用，此处所指的中断为 UART 接收中断。

bc7215_config_uart_int_en_func() 函数的输入参数为指向此中断使能函数的指针，中断使能函数必须具有如下的签名特征形式：无输入参数，无返回值：
void function_name(void);

如果配置中禁止了接收功能或者配置为不使用串口中断，则此函数不可用。

```
void bc7215_config_uart_int_dis_func(voidFunction_t);
```

配置 UART 关闭中断函数。当使用串口中断时，用户必须提供一个功能为关闭串口中断的函数。不同的控制器可能有不同的中断设计，如果 UART 有多种中断可供使用，此处所指的中断为 UART 接收中断。

bc7215_config_uart_int_dis_func() 函数的输入参数为指向此关闭中断函数的指针，关闭中断函数必须具有如下的签名特征形式：无输入参数，无返回值：
void function_name(void);

如果配置中禁止了接收功能或者配置为不使用串口中断，则此函数不可用。

```
void bc7215_config_mutex_lock_func(voidFunction_t);
```

配置互斥锁加锁函数。当使用实时多任务操作系统、且 bc7215_process_uart_data() 的呼叫和其它驱动库的函数使用不在同一个线程时，用户需要提供一个互斥锁机制，以防止资源访问的冲突。

```
void bc7215_config_mutex_unlock_func(voidFunction_t);
```

配置互斥锁解锁函数。此为上面加锁函数对应的解锁函数。

```
void bc7215_config_uart_send_byte(uint8ParamFunction_t);
```

配置 UART 单字节发送函数。用户必须提供一个 UART 发送单个字节的函数。当不是用硬件流控制时，且此函数必须在该字节完全从串口发出后才可返回。当使用硬件流控制时，则函数仅需将数据写入发送缓冲区即可返回。

bc7215_config_uart_send_byte() 函数的输入参数为指向此发送函数的函数指针。

单个字节发送函数必须具有如下的签名特征：输入参数为一个 uint8_t 类型数据，无返回值。

```
void function_name(uint8_t);
```

```
void bc7215_config_mod_set_high_funct(voidFunction_t);
```

配置 MOD 引脚置高电平函数。用户必须提供一个函数，该函数的功能为将 BC7215(A)的

MOD 置为高电平。**bc7215_config_mod_set_high_funct()** 函数的输入参数为指向此 MOD 置高函数的指针。MOD 置高电平函数必须具有如下的签名特征：无输入参数，无返回值。

```
void function_name(void);
```

```
void bc7215_config_mod_set_low_funct(voidFunction_t);
```

配置 MOD 引脚置低电平函数。用户必须提供一个函数，该函数的功能为将 BC7215(A)的

MOD 置为低电平。**bc7215_config_mod_set_low_funct()** 函数的输入参数为指向此 MOD 置低函数的指针。MOD 置低电平函数必须具有如下的签名特征：无输入参数，无返回值。

```
void function_name(void);
```

```
void bc7215_config_read_mod_funct(readIOFunction_t);
```

配置读取 MOD 状态函数。用户必须提供一个函数，该函数的功能为读取 BC7215(A)的 MOD 引脚的状态。**bc7215_config_read_mod_funct()** 函数的输入参数为指向此函数的指针。

读取 MOD 函数必须具有如下的签名特征：无输入参数，返回一 uint8_t 类型值，当 MOD 为低电平时，返回 0，当 MOD 为高电平时，返回为非 0 的值：

```
uint8_t function_name(void);
```

```
void bc7215_config_read_busy_funct(readIOFunction_t);
```

配置读取 BUSY 状态函数。当不使用串口硬件流控制时，用户必须提供一个函数，该函数的功能为读取 BC7215(A)的 BUSY 引脚的状态。**bc7215_config_read_busy_funct()** 函

数的输入参数为指向此函数的指针。读取 BUSY 函数必须具有如下的签名特征：无输入参数，返回一 uint8_t 类型值，当 BUSY 为低电平时，返回 0，当 BUSY 为高电平时，返回为非 0 的值：

```
uint8_t function_name(void);
```