RC10_LIB FrameWork用户手册

RC10_LIB将提供大量预制菜，旨在让对底层驱动不熟悉的用户也能畅快书写应用层代码。而本用户手册也是预制菜的一环，旨在让用户可以更快上手使用RC10LIB

FreeRTOS的使用

在BSP_RTOS.h文件中，封装了基本的RTOS使用，目前有基本的任务和队列

1. 目前RtosTask的任务运行拥有两种模式
    1. 超级预制菜模式：用户在初始化时候只需给定任务名，以及书写一个初始化函数用于放置start函数即可.(**注意：必须运行start函数才能注册任务，而且start函数必须在osKernelStart();之前运行,main.cpp中的**)

```
/*举例*/
/*
    1.用户需要做的，使用RtosTask实例化任务
    2.在你的初始化函数中，如此处的init()，写入start函数，指定任务的优先级、栈大小
    3.在超级预制菜模式下，只需要重写loop，写入你想执行的任务即可
*/
class FrameDemo : public RtosTask
{
public:
    FrameDemo() : RtosTask("FrameDemo") {}
    void init();
    void loop() override;
    volatile int counter = 0;
};

void FrameDemo::loop()
{
    counter++;
}

void FrameDemo::init()
{
    start(osPriorityNormal, 256);
}
```

    2. 自定义模式

```
/*举例*/
    /*
        1.用户需要做的，使用RtosTask实例化任务，同时传入构造函数的第二个参数，延迟时间(默认是1)，将其设置0，则进入自定义模式
        2.在你的初始化函数中，如此处的init()，写入start函数，指定任务的优先级、栈大小
```

```
    3.在自定义模式下，你需要自行完成任务的骨架，重写run()成员
*/
class FrameDemo : public RtosTask
{
public:
    FrameDemo() : RtosTask("FrameDemo",0) {}
    void init();
    void loop() override;
    volatile int counter = 0;
};

void FrameDemo::run()
{
    static int i;
    for(;;)
    {
        i++;
        if(i > 10)
        {
            counter++;
            i = 0;
        }

         osDelay(1);
    }

}

void FrameDemo::init()
{
    start(osPriorityNormal, 256);
}
```