CANOpen 系列教程 14

协议源码移植（二）

作者：strongerHuang

申明：该文档仅供个人学习使用

归类	CANOpen 系列教程
标签	CAN、 CANOpen、 CanFestival
网站	http://www.strongerhuang.com

目 录

一、写在前面

该系列教程上一篇文章《协议源码移植（一）》算是对 CANOpen 移植的准备工作，如果想要理解移植过程的知识，其实还有许多内容得了解。

本文挑一些重点来讲述，从协议源码添加到工程，源代码理解，到最后输出效果。

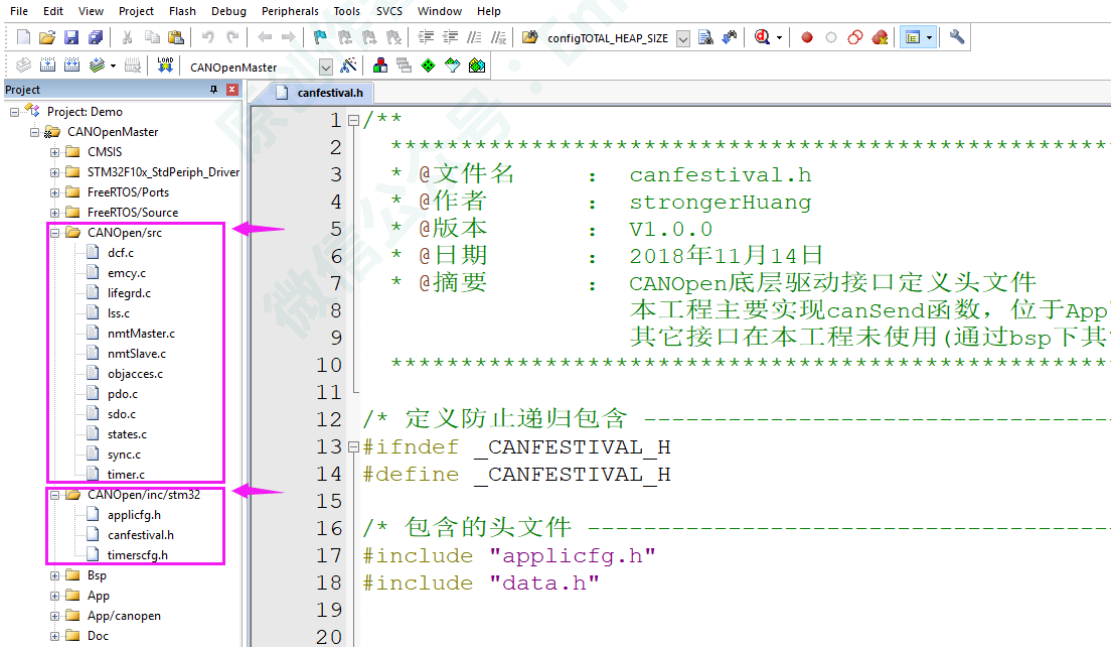
提示：该系列教程基于：CanFestival 架构、STM32F1 芯片、FreeRTOS 操作系统、Keil MDK-ARM 开发环境。

二、添加源码、路径

本节内容针对初学者做出简要描述，相关内容可参看文章《Keil 系列教程 02_新建基础软件工程》。

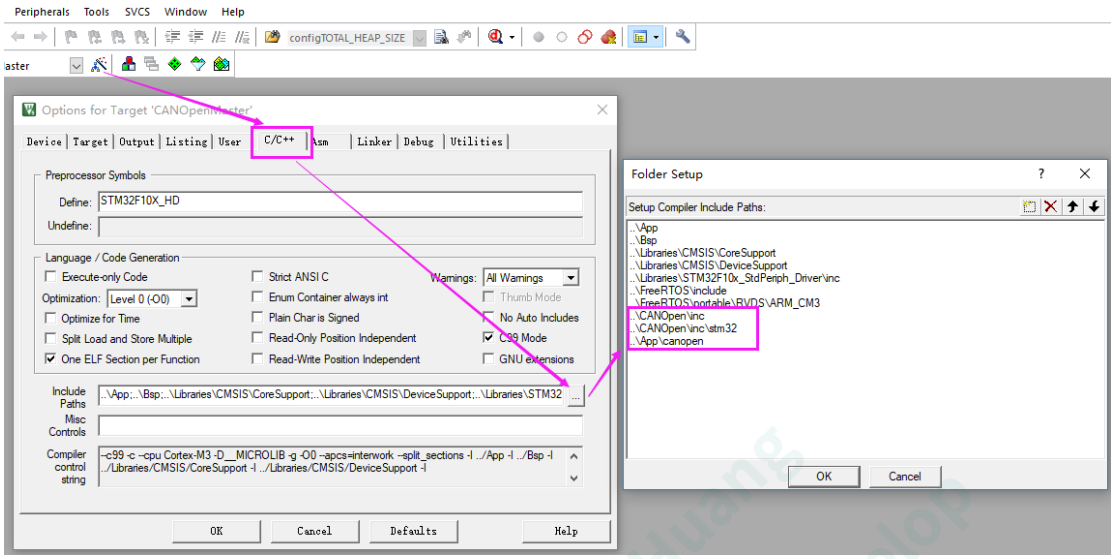
2.1 添加组、文件

简单来说，就是在你已建好的工程中添加与 CANOpen 相关的组和文件，如下图：



2.2 添加路径

添加 CANOpen 源码的 inc 头文件等相关路径，编译的时候才能找到对应文件。

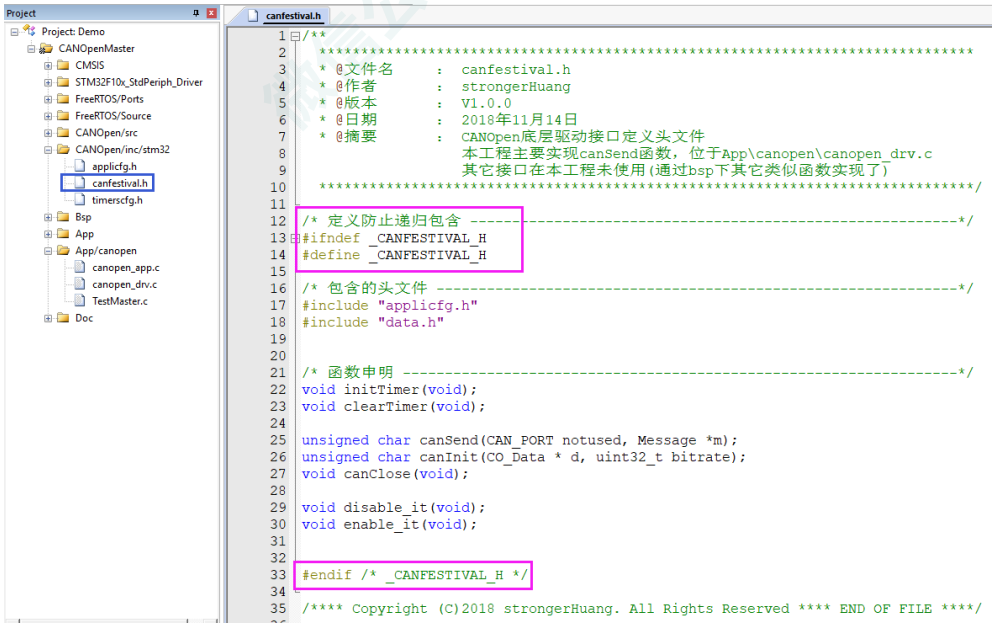


三、添加代码及分析

移植的重要过程就是添加、删除和修改源代码这一步骤。从教程上一篇文章下载，并看过源代码的朋友就会发现，其实我们需要添加的代码并不多，**主要就是需要实现几个底层的驱动函数**。

下面将重要的几点罗列出来。

1.修改 canfestival.h 文件




```
/**
  ******************************************************************************
  * @文件名     :  canfestival.h
  * @作者       :  strongerHuang
  * @版本       :  V1.0.0
  * @日期       :  2018年11月14日
  * @摘要       :  CANOpen底层驱动接口定义头文件
                    本工程主要实现canSend函数，位于App\canopen\canopen_drv.c
                    其它接口在本工程未使用(通过bsp下其它类似函数实现了)
  ******************************************************************************/

/* 定义防止递归包含 ----------------------------------------------------------*/
#ifndef _CANFESTIVAL_H
#define _CANFESTIVAL_H

/* 包含的头文件 --------------------------------------------------------------*/
#include "applicfg.h"
#include "data.h"


/* 函数申明 ------------------------------------------------------------------*/
void initTimer(void);
void clearTimer(void);

unsigned char canSend(CAN_PORT notused, Message *m);
unsigned char canInit(CO_Data * d, uint32_t bitrate);
void canClose(void);

void disable_it(void);
void enable_it(void);


#endif /* _CANFESTIVAL_H */

/**** Copyright (C)2018 strongerHuang. All Rights Reserved **** END OF FILE ****/
```

上文说了一下：**添加三行语句，防止递归包含**。

这里就是 CANOpen 定义的底层驱动接口，接口函数具体代码需要我们自己实现。

canSend 这个函数被 CANOpen 源代码调用的最多，我们最好不修改函数接口，同时需要我们实现（我在 canopen_drv.c 中实现的）。

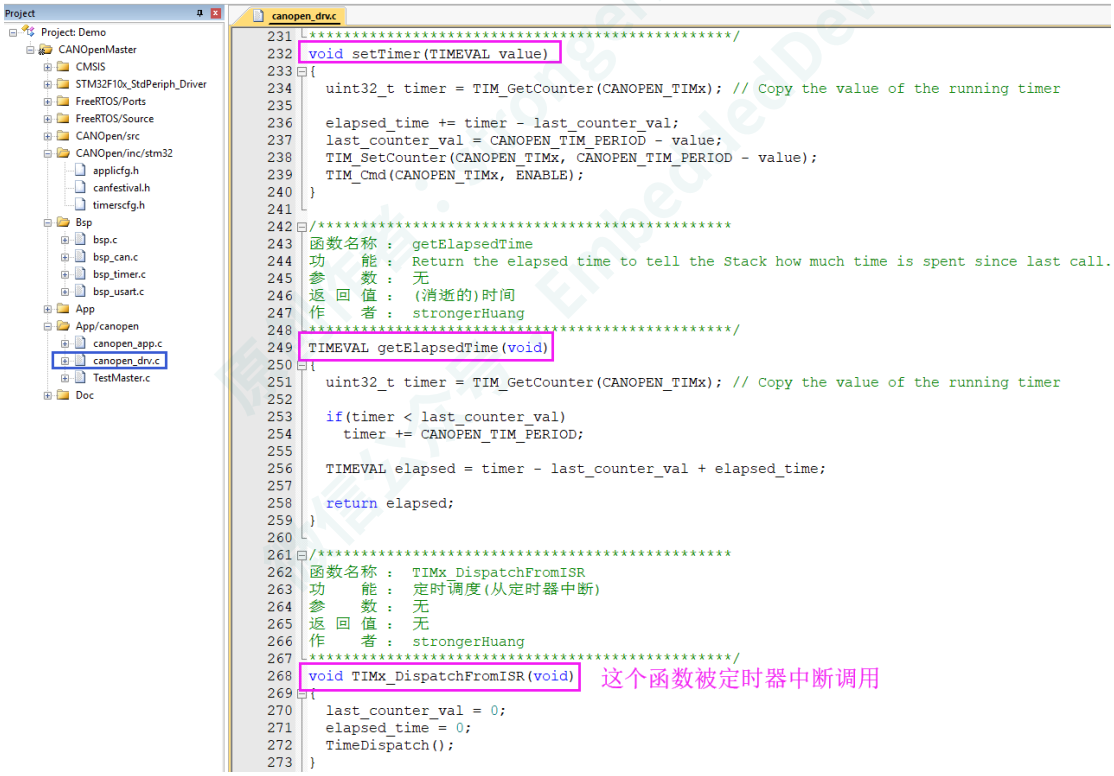
其它 **initTimer、 canInit** 等函数接口，在源码中没有调用，我也没有按照这套接口来实现（初始化我在 bsp 下实现的）。

2.底层驱动初始化代码

上面说的 initTimer、 canInit 初始化，我这边为了与我代码一致，使用 **TIM_Initializes、CAN_Initializes** 替代。

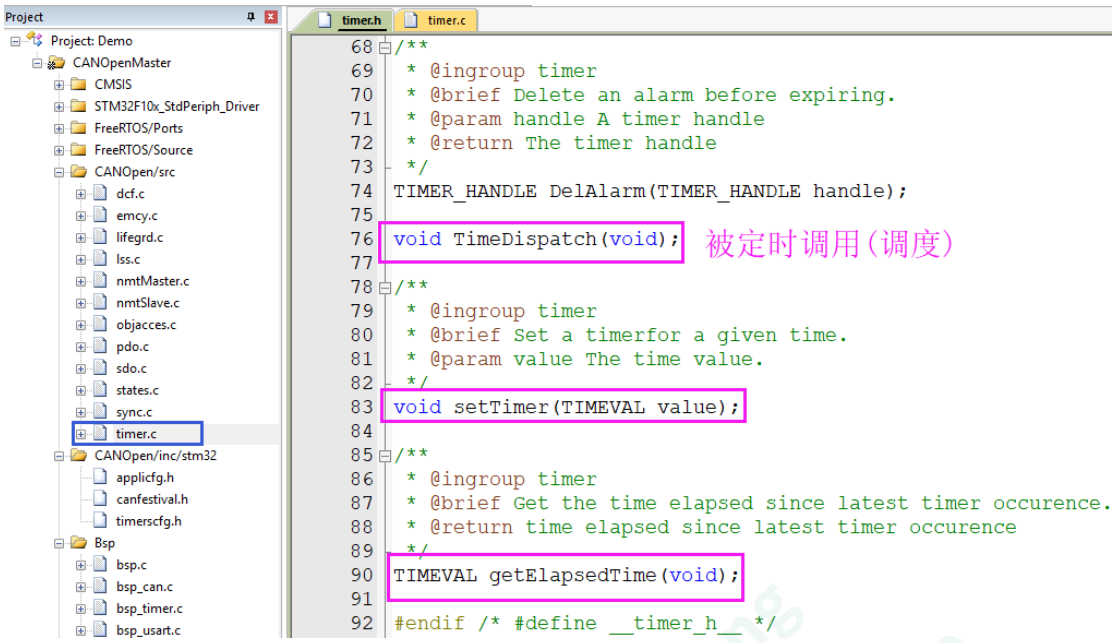
分别位于 **bsp_timer.c 和 bsp_can.c** 下面，实现的具体内容这里不描述，具体可以下载源码工程参看。

3.定时器调度相关接口



其中 **setTimer 和 getElapsedTime** 这两个函数会被 timer.c 协议源文件调用。在 timer.h 里面有什么（如下图），但函数体没有实现，需要我们自己实现。

同时，TimeDispatch 函数已经实现，定义在 timer.c，但需要被定时调用（调度）。

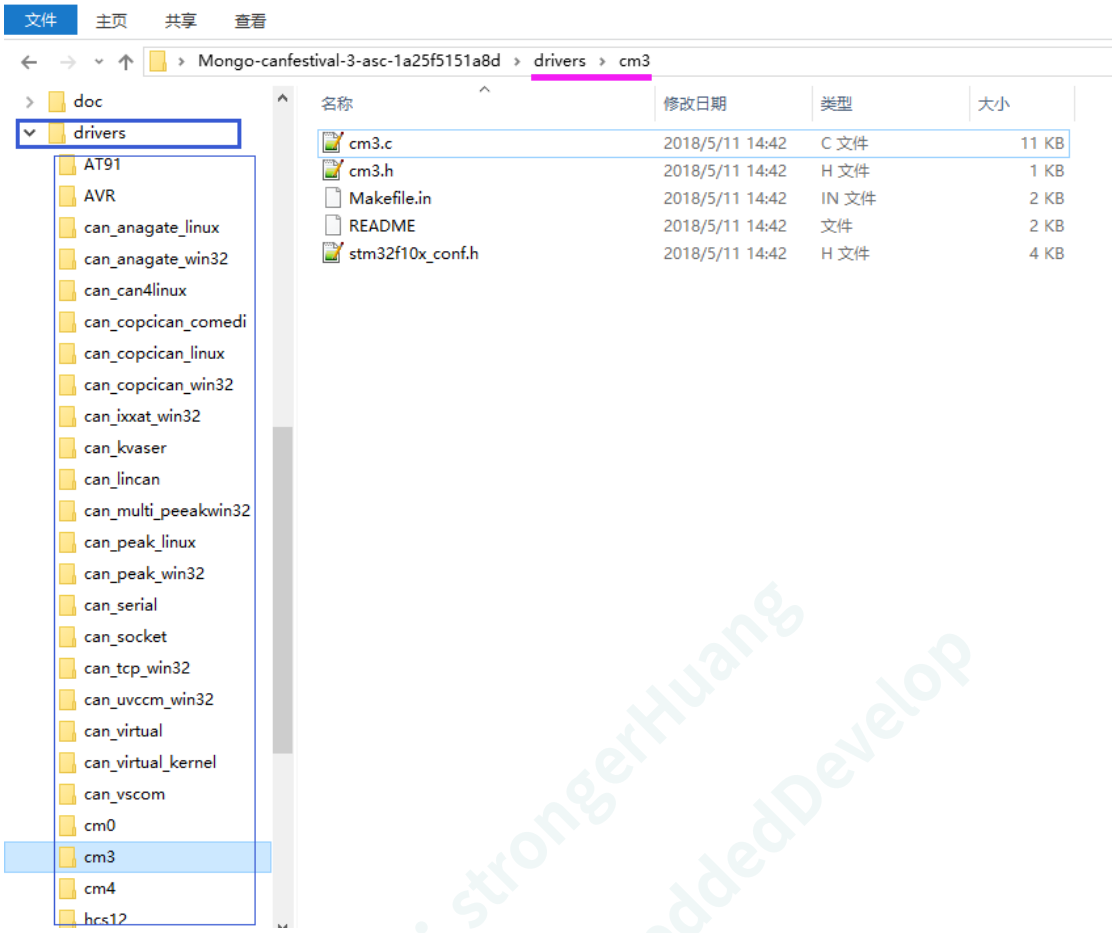


4.CAN 发送接口函数 canSend

这个 CAN 发送函数相当重要，接口最好不要自定义，因为 scr 下面多个源文件都调用了该函数。

同时，发送函数会被定时调度。所以，如果你调度方式像我例程那样，使用定时器中断的方式实现。那么，你就**要考虑在中断函数里面发送的情况**。

以上就是 CANOpen 移植，底层驱动相关的一些接口实现和说明。这部分内容，CanFestival 框架提供源码是定义在 drivers 下面，比如 STM32F1 的就是 cm3 下面的 cm3.c：



我单独提出来实现这些驱动函数是因为我跑了一个 FreeRTOS 系统。当然，你的驱动也可按照 drivers 下面那样实现。

5.其它

A.发送和接收缓存

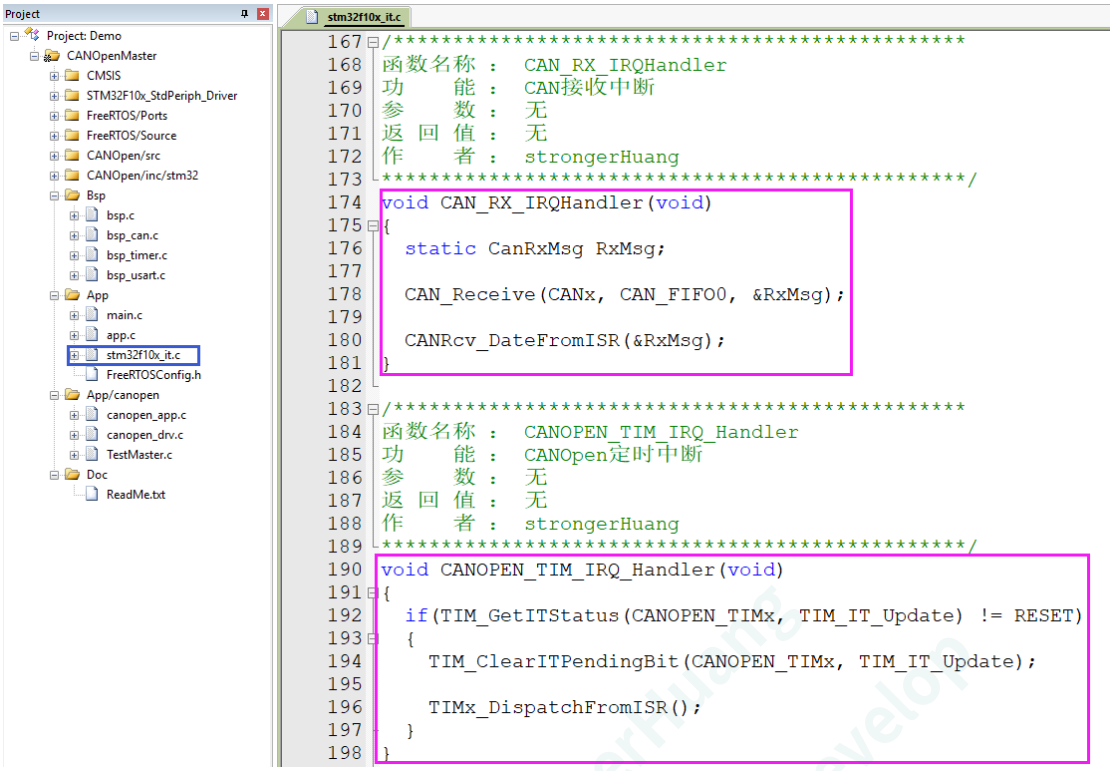
我这边是通过队列来实现发送和接收缓存，而 cm3.c 是通过 MessBuf_Write 和 MessBuf_Read 来实现缓存。

B.中断接收

我使用 CAN 中断接收数据，和定时器中断调度。

提示：我中断入口函数是宏定义实现的，需要包含宏定义头文件。

```
#define CAN_RX_IRQHandler    USB_LP_CAN1_RX0_IRQHandler
#define CANOPEN_TIM_IRQ_Handler TIM2_IRQHandler
```



C.配置节点

配置节点 ID，节点的状态，这里只是简单的 Demo，没有添加应用代码。

四、工程下载及运行效果

1.下载工程“CANOpen 工程模板（含主、从站-心跳）”

https://pan.baidu.com/s/1LzD0Epc-Z8vlHsb-sD3WVw

提取码：l2dc

提示：如果链接失效，公众号回复【CANOpen 系列教程】获取更新链接；

2.运行效果

我提供的这个 Demo 工程是一个只有心跳（间隔时间我们配置的 1000ms），所以，启动之后，你会发现**总线上间隔 1 秒有一个心跳**。

下图是我通过 CAN 分析仪抓取的 CAN 总线数据（如果你没有分析仪，可以用我系列教程 6 提供的例程，通过串口打印）

这是先启动一会儿主站，然后关闭，再启动从站的数据，重要信息我都标记出来了。

A.上线报文：ID：0x700 Data：0

B.网络管理：ID：0x000 Data：00 代表管理所有节点 Data：80 代表复位节点

C.心跳报文：ID：0x700 Data：05 代表 Operational 操作状态

提示：主站具有网络管理，而从站没有。同时，数据值（如 81、05）的含义可参看 CiA 301 手册网络管理的章节（后期进行讲述）。

五、说明

1.该文档仅供个人学习使用，版权所有，禁止商用。

2. 本文由我一个人编辑并整理，难免存在一些错误。

3.本教程收录于微信公众号「嵌入式专栏」，关注微信公众号回复【CANOpen 系列教程】即可查看全系列教程。

六、最后

我的博客：http://www.strongerhuang.com

我的 GitHub：https://github.com/EmbeddedDevelop

我的微信公众号（ID：strongerHuang）还在分享 STM8、STM32、Keil、IAR、FreeRTOS、UCOS、RT-Thread、CANOpen、Modbus...等更多精彩内容，如果想查看更多内容，可以关注我的微信公众号。