CANOpen 系列教程 13

协议源码移植（一）

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.....3

二、 移植准备.....3

2.1 硬件.....3
2.2 MCU 资源.....3
2.3 STM32 标准外设库+FreeRTOS 工程.....4
2.4 下载 Canfestival 源码并搭建好环境.....4

三、 生成对象字典.....4

四、 提取 CANOpen 源码.....6

五、 下载.....9

六、 说明.....9

七、 最后.....9

一、写在前面

基于 Canfestival 框架的 CANOpen 协议栈移植教程网上流传着许多，而且有一份不知道被复制、粘贴了多少遍的“CANOpen 移植教程”相信许多朋友都知道。

同时，百度、谷歌还能搜出许多所谓“移植代码”。其实，这许多内容，对初学者并没有多大帮助。相反，我觉得还有误导的作用。

我认为基于 Canfestival 框架的 CANOpen 协议栈**移植需要做的工作并不多**，但许多初学者无从下手，究其原因还是**有许多内容没有理解到位**。

本教程站在初学者角度，尽量将重要内容讲述到位，后面提供移植好的 Demo 供大家参考学习。

二、移植准备

移植前，先让大家认识一下移植的一些准备条件。

2.1 硬件

两块带有 MCU、CAN 控制器和收发器的板卡。

该移植教程以 STM32F1（自带 CAN 控制器），带有 CAN 收发器的开发板为例来讲述。

条件允许的情况下，可以购买一个 CAN 总线分析仪。没有分析仪的朋友，可以用我《CANOpen 系列教程 06》提供的一个例程来抓取 CAN 总线数据。

因例程在中断里用串口打印传输数据。所以，用例程抓取的 CAN 总线传输速率不能太快（建议低于 20 帧/秒）。

2.2 MCU 资源

1.CAN：CAN 总线通信（**必备**）；
2.TIM：CANOpen 协议调度（**必备**）；
3.UART：调试信息（选配）；
4.GPIO：板卡状态指示灯（选配）；

2.3 STM32 标准外设库+FreeRTOS 工程

本教程基于 STM32 标准外设库和 FreeRTOS 系统搭建的工程为例（很早之前我分享过），这里不讲述。

运行 RTOS 主要是牵涉到 CAN 总线数据的发送和接收需要两个线程来处理，以及增加一个 CANOpen 应用程序线程。

基于其他 MCU 以及 RTOS 原理类似，也可参考本文。

2.4 下载 Canfestival 源码并搭建好环境

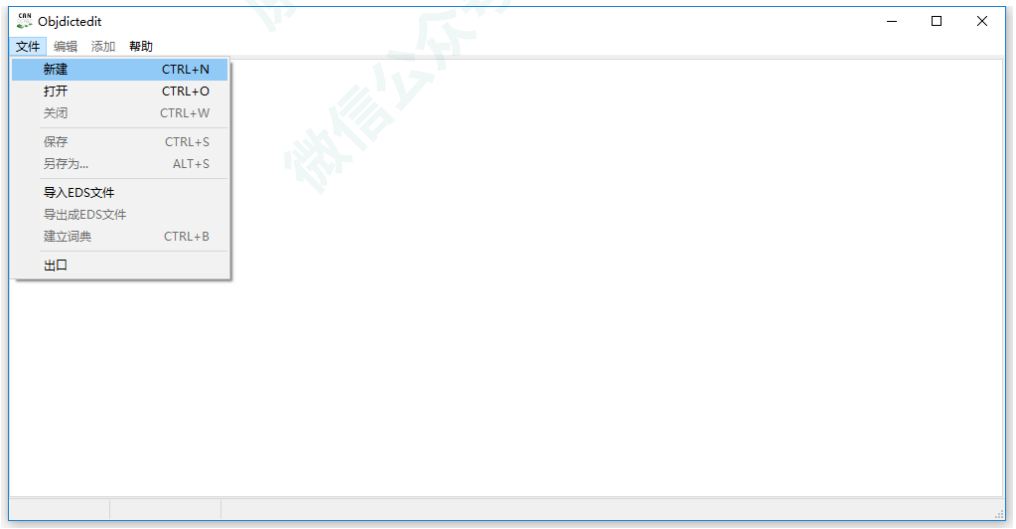
这里的内容，如果你不明白，请务必先阅读该教程前几篇相关文章。

三、生成对象字典

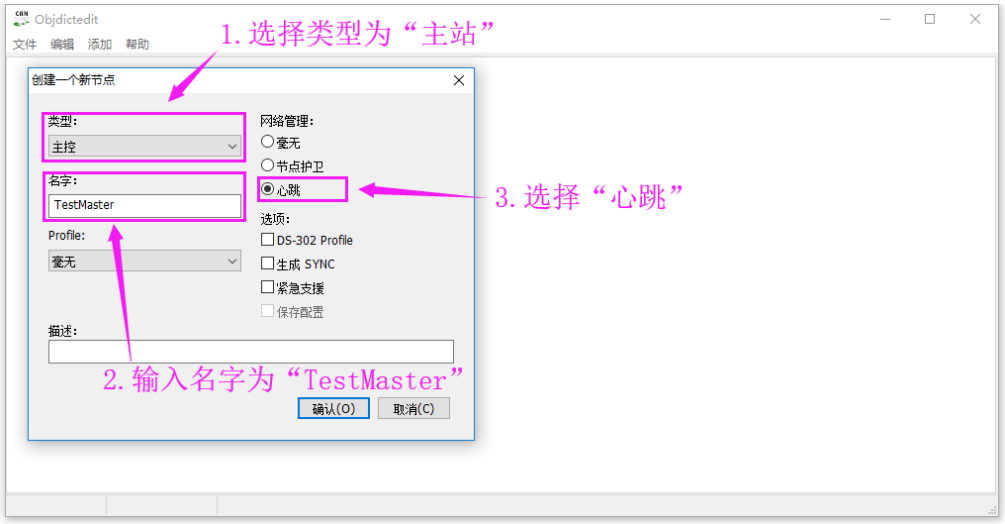
本移植教程重点是将 CANOpen 协议源码移植到 MCU 中，使其成功运行。因此，对象字典只配置最基础的“心跳”。

下面教大家在对象字典编辑器中生成带“心跳”的对象字典。

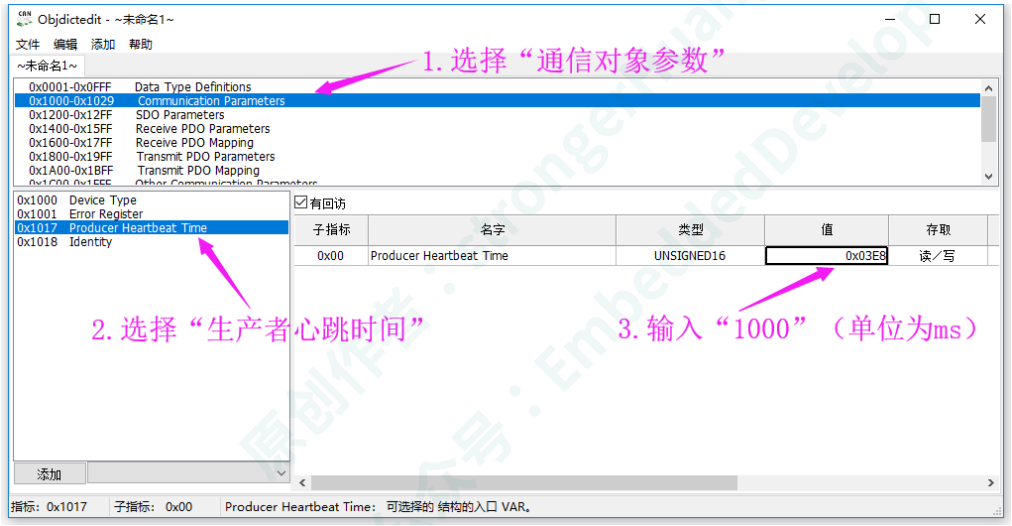
1.打开编辑器，文件 -> 新建



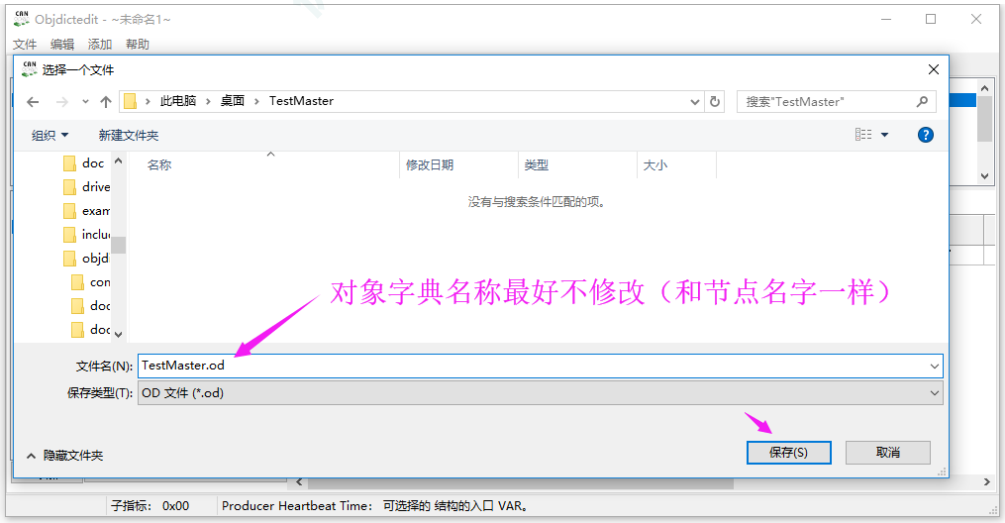
2.新建“主站”节点



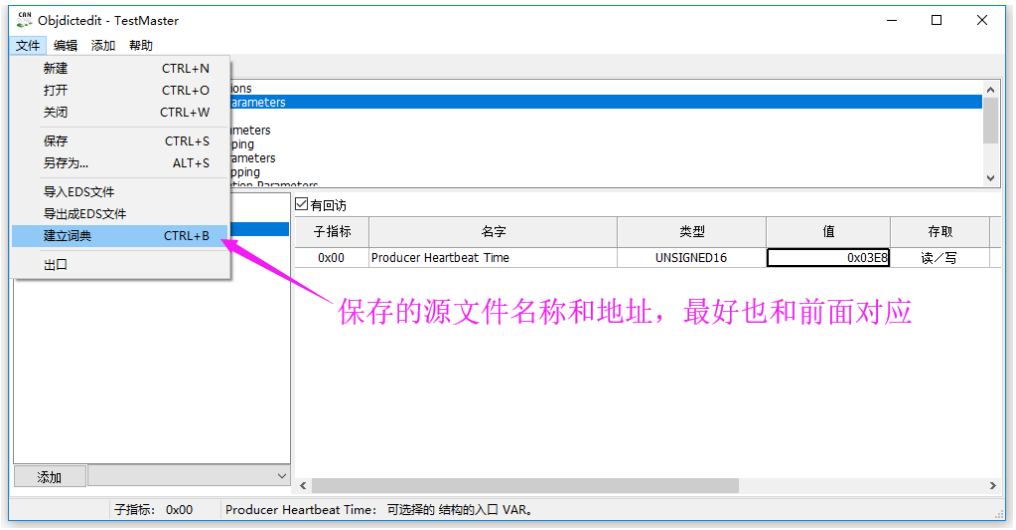
3.配置心跳信息



4.保存对象字典：文件 -> 保存 -> 保存在一个指定位置



5.建立对象字典



到这里就生成了三个文件：**TestMaster.od、 TestMaster.c 和 TestMaster.h** 需保存好，在后面需要使用这个对象字典源码。

提示：

A.上面牵涉到三个地方保存名字：节点名字、对象字典工程名字、对象字典源文件名字，建议都一样。

B.主站和从站的生成原理一样，上面是生成主站 TestMaster 对象字典，从站 TestSlave 对象字典请按照同样方式生成即可。

四、提取 CANOpen 源码

本节内容主要针对初学者使其更加理解移植的代码工程，将 CanFestival 中源代码提取做一定说明，并且尽量将源码文件对应到自己工程下。

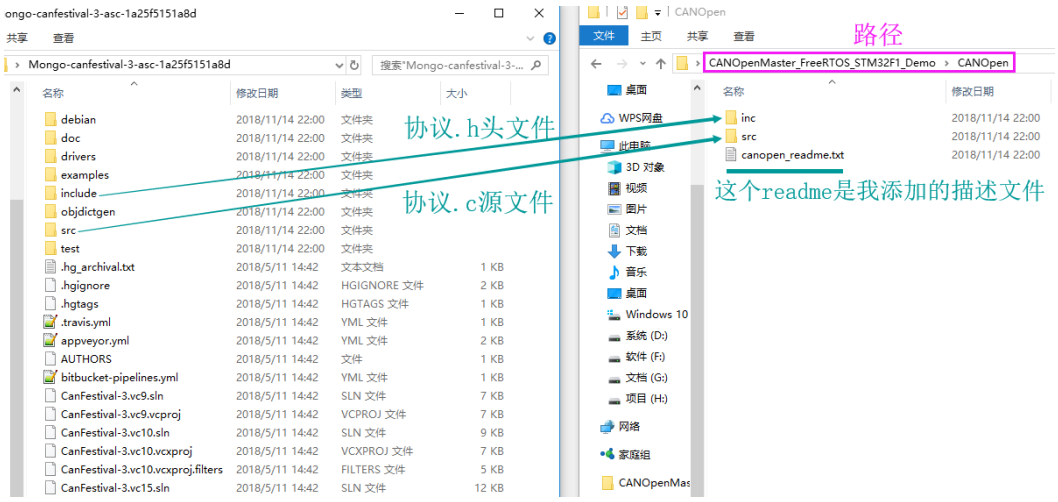
当然，本文按照常规方式提取，你也可以按照自己方式提取。同时，整个工程下其它，如 STM32 标准外设库、FreeRTOS 这些文件在这里不作说明。

1.提取头、源文件（如下图）

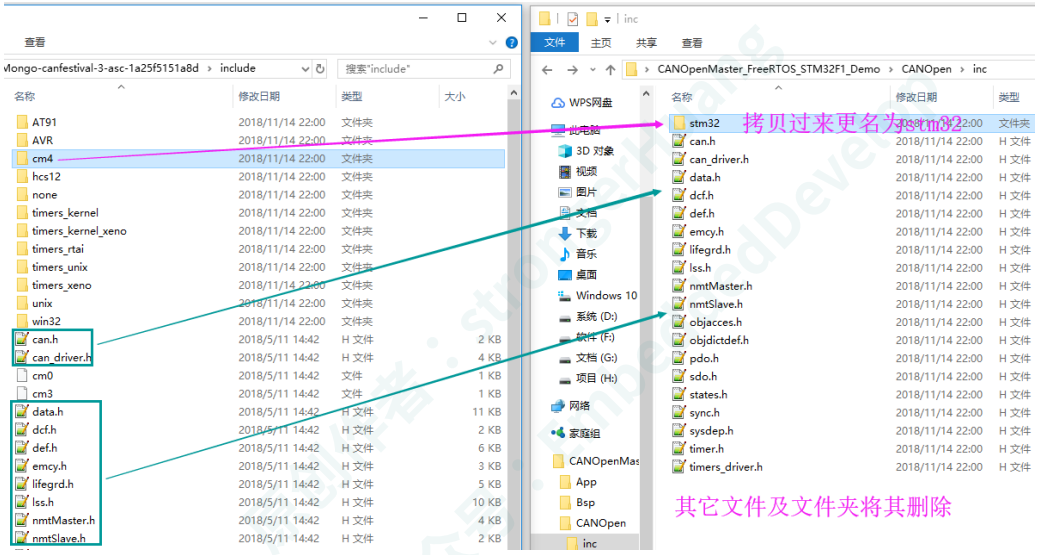
CANOpen\inc 目录下 19 个.h 头文件，来自 Canfestival->include 下目录 19 个头文件；

CANOpen\inc\stm32 目录下 3 个.h 头文件来自 Canfestival->include\cm4；
其中 canfestival.h 文件是函数接口定义（声明），函数内容需要自己实现（位于：App\canopen 目录下 canopen_drv.c）；

CANOpen\src 目录下 12 个.c 源文件，来自 Canfestival->src 目录 12 个源文件（symbols.c 源文件为 linux 下使用的文件，不需要提取）；
其中需要删除 dcf.c 文件下第 59、98 行前面的“inline”关键字；



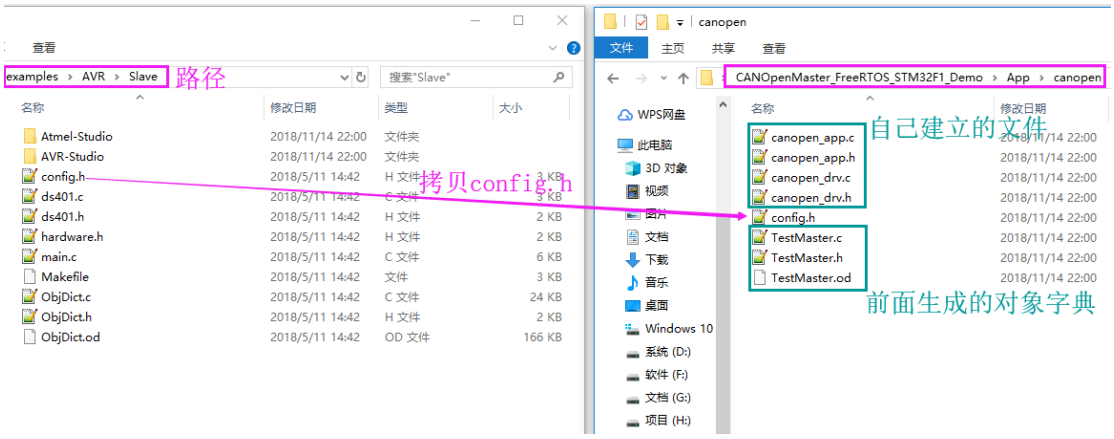
2.头文件说明



要修改一下其中的 canfestival.h 文件：


```
 * @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 ****/
```

3.建立 canopen 文件夹并提取文件



在工程下建立 canopen 文件夹，提取相应的文件，同时建立自己的文件：canopen_app 和 canopen_drv.

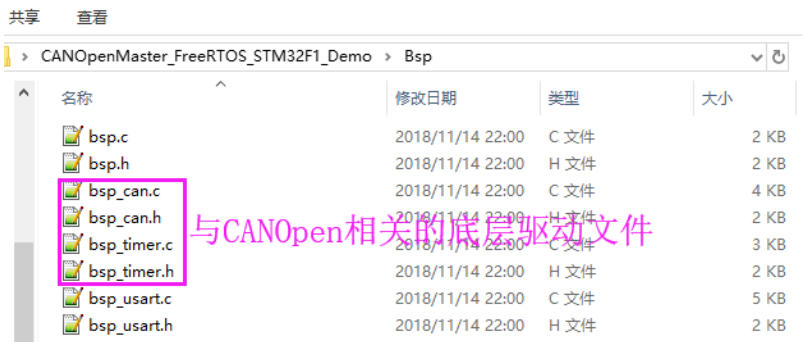
还需要修改一下 congfig.h 文件，如下图：

```
#else   // GCC
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/sleep.h>
#include <avr/wdt.h>
#endif  // GCC

//#define WD_SLEEP
// Needed defines by Atmel lib
#define FOSC            8000        // 16 MHz External cristal
#ifndef F_CPU
#define F_CPU           (1000UL*FOSC) // Need for AVR GCC
#endif
#define CAN_BAUDRATE    250

// Needed defines by Canfestival lib
#define MAX_CAN_BUS_ID 1
#define SDO_MAX_LENGTH_TRANSFER 32

#define SDO_MAX_SIMULTANEOUS_TRANSFERS 1
#define NMT_MAX_NODE_ID 128
#define SDO_TIMEOUT_MS 3000U
#define MAX_NB_TIMER 8

// CANOPEN_BIG_ENDIAN is not defined
#define CANOPEN_LITTLE_ENDIAN 1

#define US_TO_TIMEVAL_FACTOR 8

#define REPEAT_SDO_MAX_SIMULTANEOUS_TRANSFERTS_TIMES(repeat)\
repeat
```

```
#else   // GCC
//#include <inttypes.h>
//#include <avr/io.h>
//#include <avr/interrupt.h>
//#include <avr/pgmspace.h>
//#include <avr/sleep.h>
//#include <avr/wdt.h>
#endif  // GCC

//#define WD_SLEEP
// Needed defines by Atmel lib
#define FOSC            8000        // 16 MHz External cristal
#ifndef F_CPU
#define F_CPU           (1000UL*FOSC) // Need for AVR GCC
#endif
#define CAN_BAUDRATE    250

// Needed defines by Canfestival lib
#define MAX_CAN_BUS_ID 1
#define SDO_MAX_LENGTH_TRANSFER 32
#define SDO_BLOCK_SIZE 16
#define SDO_MAX_SIMULTANEOUS_TRANSFERS 1
#define NMT_MAX_NODE_ID 128
#define SDO_TIMEOUT_MS 3000U
#define MAX_NB_TIMER 8

// CANOPEN_BIG_ENDIAN is not defined
#define CANOPEN_LITTLE_ENDIAN 1

#define US_TO_TIMEVAL_FACTOR 8

#define REPEAT_SDO_MAX_SIMULTANEOUS_TRANSFERS_TIMES(repeat)\
repeat
```

1. 注释不用包含
2. 添加宏定义
3. 修改名称

4.建立 CAN 和 TIM 底层驱动文件



在上面 canfestival.h 文件提供了 CAN 和 TIM 的驱动接口，但我们这里不用它那一套接口，自己定义在 bsp_can 和 bsp_timer 源代码下。

五、下载

为照顾初学者，将移植过程讲述的更加详细，移植工作在下一篇文章还会继续讲述，提前把移植好的、带有“心跳”的 Demo 工程给大家下载。

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；

六、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。

七、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。