CANOpen 系列教程 09

CANOpen 对象字典

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

原创作者：strongerHuang 微信公众号：EmbeddedDevelop

版权所有：**禁止商用**

Copyright @2018 strongerHuang

目 录

原创作者：strongerHuang
微信公众号：EmbeddedDevelop

一、写在前面

上一篇文章主要是引导大家学习 CANOpen 协议通信接口的相关内容。通信接口的内容有点多，对于初学者来说，如果看不懂，请多看几遍（最好有个印象）。

CANopen 设备最重要的一个部分就是对象字典。对象字典本质上是一种被预先安排的对象序列，可通过网络访问。**字典里面的每一对象均可使用一个 16-bit 主索引和 8-bit 子索引寻址到**。

初学者学习如果难理解通信接口（上一篇）有些内容，如：PDO、SDO 及网络管理对象，可以结合对象字典索引来学习和记忆。

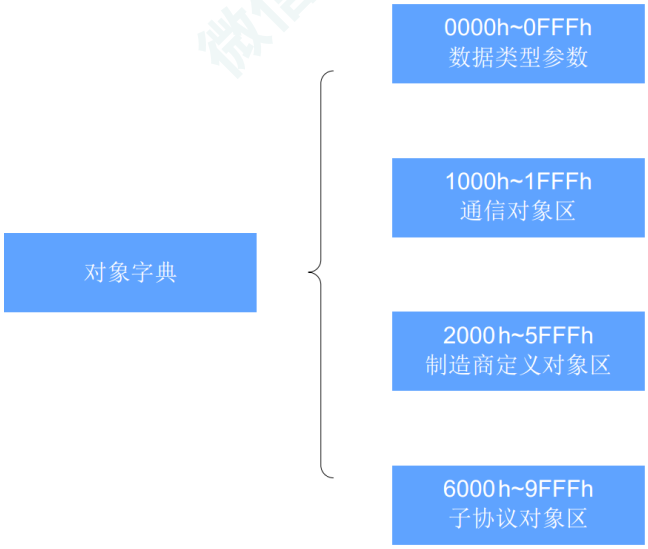
本文章收录于【CANOpen 系列教程】，在我的博客分类“CANOpen 系列教程”也能查找到。

为了方便大家平时公交、地铁、外出办事也能用手机随时随地查看该教程，该系列教程也同步更新于微信公众号【EmbeddedDevelop】，关注微信公众号回复【CANOpen 系列教程】即可查看。

二、对象字典重要内容

对象字典 OD：Object dictionary。

一组带有 16 位索引和 8 位子索引的数据或对象。对象字典下面主要包含：**数据类型、通信对象、应用对象**。



对象字典主索引：

索引	对象
000	未使用
0001h—001Fh	静态数据类型 (标准数据类型，如 Boolean、Integer16)
0020h—003Fh	复杂数据类型 (预定义由简单类型组合成的结构如 PDOCommPar、 SDOParmeter)
0040h—005Fh	制造商规定的复杂数据类型
0060h—007Fh	设备子协议规定的静态数据类型
0080h—009Fh	设备子协议规定的复杂数据类型
00A0h—0FFFh	保留
1000h —1FFFh	通信子协议区域 (如设备类型，错误寄存器，支持的 PDO 数量)
2000h —5FFFh	制造商特定子协议区域 (如功能码映射)
6000h —9FFFh	标准的设备子协议区域 (如 DSP-402 协议)
A000h—FFFFh	保留

2.1 通信子协议区

通信子协议区，如上图 1000h 到 1FFFh 这个区域。也就是通信接口（或通信对象）协议区。

对于初学者来说，**通信对象子协议区**是比较重要的一个区域，牵涉到上一篇文章《**通信接口**》讲述的大部分内容。同时，这之间的关系也非常紧密。

通讯对象子协议区：

Index range 索引范围	Description 描述
1000_h to 1029_h	General communication objects 通用通讯对象
1200_h to $12FF_h$	SDO parameter objects SDO 参数对象
1300_h to $13FF_h$	CANopen safety objects 安全对象
1400_h to $1BFF_h$	PDO parameter objects PDO 参数对象
$1F00_h$ to $1F11_h$	SDO manager objects SDO 管理对象
$1F20_h$ to $1F27_h$	Configuration manager objects 配置管理对象
$1F50_h$ to $1F54_h$	Program control object 程序控制对象
$1F80_h$ to $1F89_h$	NMT master objects 网络管理主机对象

如上图，通讯对象子协议区又划分为几个区域，而其中的 1000h 到 1029h 为**通用通讯对象**。

通用通讯对象里面的内容已经被约定好，如果我们需要用到，可直接配置使用即可。

通用通讯对象内容：

Index 索引	Object 对象	Name 名字
1000_h	VAR 变量	Device type 设备类型
1001_h	VAR 变量	Error register 错误寄存器
1002_h	VAR 变量	Manufacturer status register 制造商状态寄存器
1003_h	ARRAY 数组	Pre-defined error field 预定义错误场
1005_h	VAR 变量	COB-ID Sync message 同步报文 COB 标识符
1006_h	VAR 变量	Communication cycle period 同步通信循环周期（单位 us）
1007_h	VAR 变量	Synchronous windows length 同步窗口长度(单位 us)
1008_h	VAR 变量	Manufacturer device name 制造商设备名称
1009_h	VAR 变量	Manufacturer hardware version 制造商硬件版本
$100A_h$	VAR 变量	Manufacturer software version 制造商软件版本
$100C_h$	VAR 变量	Guard time 守护时间（单位 ms）
$100D_h$	VAR 变量	Life time factor 寿命因子（单位 ms）
1010_h	VAR 变量	Store parameters 保存参数
1011_h	VAR 变量	Restore default parameters 恢复默认参数
1012_h	VAR 变量	COB-ID time stamp 时间报文 COB 标识符（发送网络时间）
1013_h	VAR 变量	High resolution time stamp 高分辨率时间标识
1014_h	VAR 变量	COB-ID emergency 紧急报文 COB 标识符
1015_h	VAR 变量	Inhibit time emergency 紧急报文禁止时间（单位 100us）
1016_h	ARRAY 数组	Consumer heartbeat time 消费者心跳时间间隔(单位 ms)
1017_h	VAR 变量	Producer heartbeat time 生产者心跳时间间隔（单位 ms）
1018_h	RECORD 记录	Identity object 厂商 ID 标识对象
1019_h	VAR 变量	Sync.counter overflow value 同步计数溢出值
1020_h	ARRAY 数组	Verify configuration 验证配置
1021_h	VAR 变量	Store EDS 存储 EDS
1022_h	VAR 变量	Storage format 存储格式
1023_h	RECORD 记录	OS command 操作系统命令
1024_h	VAR 变量	OS command mode 操作系统命令模式
1025_h	RECORD 记录	OS debugger interface 操作系统调试接口
1026_h	ARRAY 数组	OS prompt 操作系统提示
1027_h	ARRAY 数组	Module list 模块列表
1028_h	ARRAY 数组	Emergency consumer 紧急报文消费者
1029_h	ARRAY 数组	Error behavior 错误行为

通信子协议区的内容非常重要，也比较多，需要拆分并一个一个掌握。可结合周立功提供相关教程，以及前面提到的 CiA 301 手册理解。

我这里就不一一列出来了，简单再列一个**通信子协议区中 PDO 通信参数（对象）与映射参数：**

名称		COB-ID	通信对象	映射对象
RPDO	1	200h + Node_ID	1400h	1600h
	2	300h + Node_ID	1401h	1601h
	3	400h + Node_ID	1402h	1602h
	4	500h + Node_ID	1403h	1603h
TPDO	1	180h + Node_ID	1800h	1A00h
	2	280h + Node_ID	1801h	1A01h
	3	380h + Node_ID	1802h	1A02h
	4	480h + Node_ID	1803h	1A03h

写到这里，想必很多朋友都能理解，也能明白这些内容。更多的内容，可以结合这种拆分的思路去理解。

三、对象字典生成工具

上面介绍了对象字典的一些内容，可能还是有很多初学者不明白。同时，也就可能会产生疑问：这个对象字典我们需要字典和我们代码有什么关系呢？

可以简单来说，就是一些 16 位索引和 8 位子索引对应的变量数据，来下看下代码（下面两图），加深理解：

```
const indextable Test_objdict[] =
{
  { (subindex*)Test_Index1000,sizeof(Test_Index1000)/sizeof(Test_Index1000[0]), 0x1000},
  { (subindex*)Test_Index1001,sizeof(Test_Index1001)/sizeof(Test_Index1001[0]), 0x1001},
  { (subindex*)Test_Index1018,sizeof(Test_Index1018)/sizeof(Test_Index1018[0]), 0x1018},
  { (subindex*)Test_Index1200,sizeof(Test_Index1200)/sizeof(Test_Index1200[0]), 0x1200},
  { (subindex*)Test_Index1400,sizeof(Test_Index1400)/sizeof(Test_Index1400[0]), 0x1400},
  { (subindex*)Test_Index1401,sizeof(Test_Index1401)/sizeof(Test_Index1401[0]), 0x1401},
  { (subindex*)Test_Index1402,sizeof(Test_Index1402)/sizeof(Test_Index1402[0]), 0x1402},
  { (subindex*)Test_Index1403,sizeof(Test_Index1403)/sizeof(Test_Index1403[0]), 0x1403},
  { (subindex*)Test_Index1600,sizeof(Test_Index1600)/sizeof(Test_Index1600[0]), 0x1600},
  { (subindex*)Test_Index1601,sizeof(Test_Index1601)/sizeof(Test_Index1601[0]), 0x1601},
  { (subindex*)Test_Index1602,sizeof(Test_Index1602)/sizeof(Test_Index1602[0]), 0x1602},
  { (subindex*)Test_Index1603,sizeof(Test_Index1603)/sizeof(Test_Index1603[0]), 0x1603},
  { (subindex*)Test_Index1800,sizeof(Test_Index1800)/sizeof(Test_Index1800[0]), 0x1800},
  { (subindex*)Test_Index1801,sizeof(Test_Index1801)/sizeof(Test_Index1801[0]), 0x1801},
  { (subindex*)Test_Index1802,sizeof(Test_Index1802)/sizeof(Test_Index1802[0]), 0x1802},
  { (subindex*)Test_Index1803,sizeof(Test_Index1803)/sizeof(Test_Index1803[0]), 0x1803},
  { (subindex*)Test_Index1A00,sizeof(Test_Index1A00)/sizeof(Test_Index1A00[0]), 0x1A00},
  { (subindex*)Test_Index1A01,sizeof(Test_Index1A01)/sizeof(Test_Index1A01[0]), 0x1A01},
  { (subindex*)Test_Index1A02,sizeof(Test_Index1A02)/sizeof(Test_Index1A02[0]), 0x1A02},
  { (subindex*)Test_Index1A03,sizeof(Test_Index1A03)/sizeof(Test_Index1A03[0]), 0x1A03},
};
```

```
/* index 0x1200 :   Server SDO Parameter. */
UNS8 Test_highestSubIndex_obj1200 = 2; /* number of subindex - 1*/
UNS32 Test_obj1200_COB_ID_Client_to_Server_Receive_SDO = 0x600; /* 1536 */
UNS32 Test_obj1200_COB_ID_Server_to_Client_Transmit_SDO = 0x580;  /* 1408 */
subindex Test_Index1200[] =
 {
   { RO, uint8, sizeof (UNS8), (void*)&Test_highestSubIndex_obj1200, NULL },
   { RO, uint32, sizeof (UNS32), (void*)&Test_obj1200_COB_ID_Client_to_Server_Receive_SDO, NULL },
   { RO, uint32, sizeof (UNS32), (void*)&Test_obj1200_COB_ID_Server_to_Client_Transmit_SDO, NULL }
 };
/* index 0x1400 :   Receive PDO 1 Parameter. */
UNS8 Test_highestSubIndex_obj1400 = 6; /* number of subindex - 1*/
UNS32 Test_obj1400_COB_ID_used_by_PDO = 0x200;  /* 512 */
UNS8 Test_obj1400_Transmission_Type = 0x0;  /* 0 */
UNS16 Test_obj1400_Inhibit_Time = 0x0;  /* 0 */
UNS8 Test_obj1400_Compatibility_Entry = 0x0;  /* 0 */
UNS16 Test_obj1400_Event_Timer = 0x0; /* 0 */
UNS8 Test_obj1400_SYNC_start_value = 0x0; /* 0 */
subindex Test_Index1400[] =
 {
   { RO, uint8, sizeof (UNS8), (void*)&Test_highestSubIndex_obj1400, NULL },
   { RW, uint32, sizeof (UNS32), (void*)&Test_obj1400_COB_ID_used_by_PDO, NULL },
   { RW, uint8, sizeof (UNS8), (void*)&Test_obj1400_Transmission_Type, NULL },
   { RW, uint16, sizeof (UNS16), (void*)&Test_obj1400_Inhibit_Time, NULL },
   { RW, uint8, sizeof (UNS8), (void*)&Test_obj1400_Compatibility_Entry, NULL },
   { RW, uint16, sizeof (UNS16), (void*)&Test_obj1400_Event_Timer, NULL },
   { RW, uint8, sizeof (UNS8), (void*)&Test_obj1400_SYNC_start_value, NULL }
 };
```

3.1 对象字典的代码需要自己写吗？

很多初学者，包括我（之前初学时）也会产生疑问：对象字典的代码需要我们自己写吗？

答案：不用自己写。

当然，对象字典的代码可以自己写，但完全没必要。1.自己写不一定正确；2.有现成工具可以节约大量时间。

3.2 Canfestival 对象字典生成工具

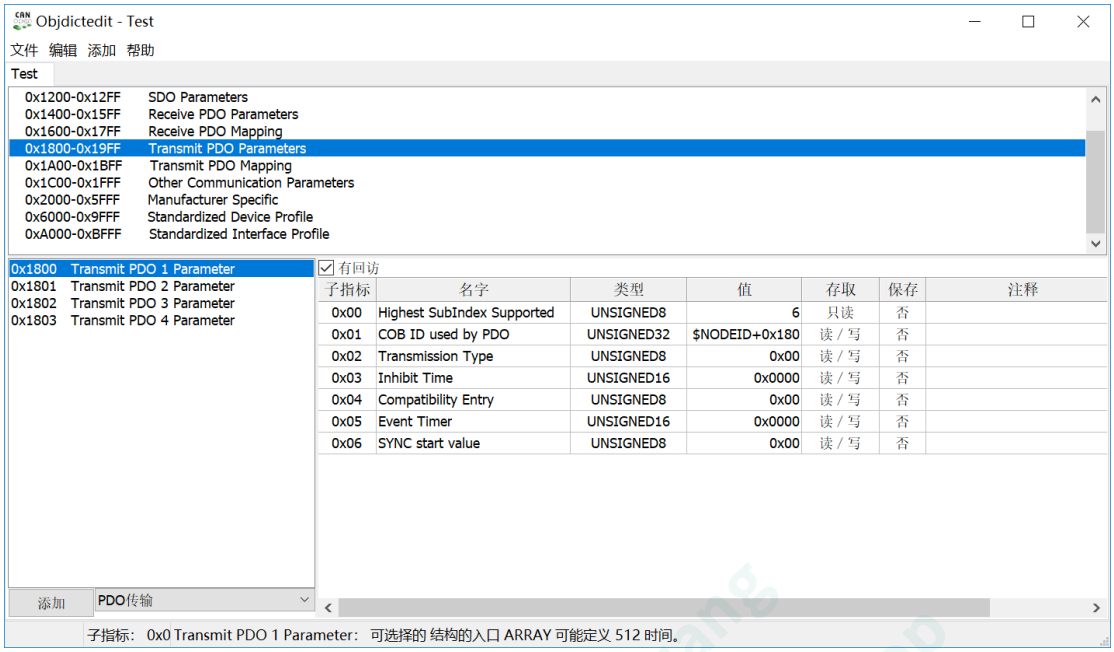
我微信公众号分享《CANOpen 系列教程》的封面中有一行英文：Canfestival. 相信学过 CANOpen 的朋友都应该知道我接下来会结合这套 **Canfestival 免费开源的 CANOpen 架构**来讲述。

其中，Canfestival 里面包含我们需要的 **CANOpen 协议源码**和**对象字典生成工具**。

有了对象字典生成工具，对象字典的代码就可以通过它来自动生成。搭建 Canfestival 对象字典生成工具的环境是一个重点。

对于很多初学者来说搭建环境也是一个难点，不少初学者就卡在这一关，然后，学习 CANOpen 就没有继续下去了。

简单的说，搭建对象字典生成工具的环境步骤不多，可能会因为诸多因素导致搭建失败。搭建成功，运行之后会出现如下图界面：



在学习对象字典生成工具时，会看见一个词：EDS，**即 Electronic Data Sheet 电子数据单**。

像周立功的 CANOpen 从站模块就配有工具生成 EDS 文件。当然，Canfestival 提供的工具也能导出 EDS 文件。

Canfestival 对象字典生成工具搭建过程，及相关内容我将在下一篇文章详细讲述。

四、说明

1.该文档部分内容来自网络，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。

五、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。

原创作者：strongerHuang 微信公众号：EmbeddedDevelop