1.前言

GoogolTech EtherCAT 配置文件主要包含两种格式的文件，一种是后缀名为“.eni”格式的，此配置文件用于描述运行时 EtherCAT 总线上所有的从站配置和总线的一些配置参数，此文件经解析后被协议栈使用，以下简称 ENI 文件；另外一种是后缀名为“.sii”格式的，此格式本质上是解析了单个标准 EtherCAT 从站配置文件 xml 的一种简化版从站配置文件，用于 GoogolTech 内部配置工具和协议栈等的使用，以下简称 SII 文件。从结构上来讲，可以理解为 ENI 文件是多个 SII 文件和特定总线参数配置的集合。为了软件上的兼容性，SII 文件和 ENI 文件的文件结构是基本相同的，所不同的只是各自所包含的内容。下面的章节先介绍两种配置文件的相同部分，然后再分别介绍用于从站描述的 SII 文件和总线控制器使用的 ENI 文件具体结构。

2.配置文件的元结构

EtherCAT 配置文件的元结构是以“行”为单位的字符串描述形式，为了表达文件层级式结构，配置文件有两种行结构，一种：Section 行，此行用于表达某一段参数配置的标识，意即下面一系列参数配置属于这一个 Section；第二种：Parameter 行，此行用于表达某一组参数的配置，可以在一行种配置多个参数，参数的具体数值也配置在 Parameter 行中。为了技术人员更好的阅读配置文件，两种类型的行中设定了若干 tags（特定字符串），以自然语言的形式描述该行的特性。

1）.Tags

目前支持 tag 为以下字符串：
"CommonInfo","MasterInfo","SlaveIndex","Param","name","Section","Value","sync","PDO"，"entry","dc","iomap","startsdo","DWORD","HEX","STRING","FLOAT","info","SlaveStart", "SlaveEnd".

以上所有 tag 在配置文件中是不包含“”的。对这些 Tag 可以这样理解：以上支持的 tag 在配置文件可以被软件识别，除此之外所有的字符串在配置文件中出现都是不合法的，强行添加可能会引起软件错误。

另外为了方便技术人员记录一些额外信息，配置文件的解析软件支持在配置文件中写入一些与配置无关的 comment，两种形式都可以：（1）在某一空白行起始位置以“//”开始，在“//”之后写入一些注释信息等，此行为注释行，不属于 Section 行或 Parameter 行；（2）在 Section 行或者 Parameter 行结尾处，以“//”开始，写入一些注释信息，“//”之后的信息都不会被软件解析。

2）.Section 行

范例：　　[slave_sync2_PDO_entry]

以上为一个典型的 Section 行范例，基本包含有一个 Section 行的所有内容。以上范例的

结构可以用下图表示：

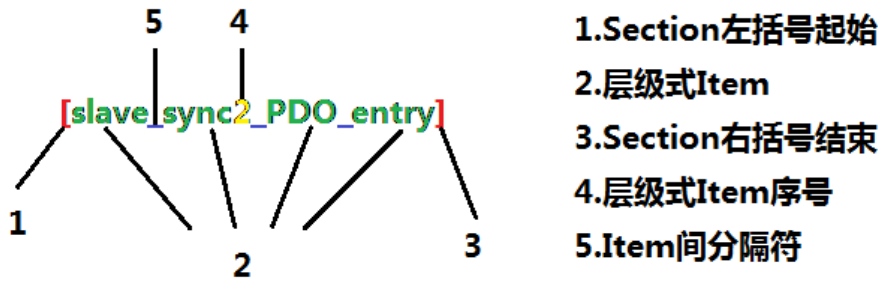

图 2-1　Section 行结构图

一个 Section 行的内容都包含在一对方括号中，方括号中可以多个 Item，以下划线连接在一起，用以表达树形的文件结构，层级式 Item 的最多级数为 5 级，基本上 5 级已经可以表达比较复杂的文件数据结构了；每一个 Item 的名称都来自于上一小节中的 tag，系统不认识除上节 tag 外的 Item 名称，对于 Parameter 行也是这样。另外，上图中 4 表示的是某一级 Item 的 Index，每一级的 Item 都可以有 Index，Index 的有无根据你要描述的文件结构而定，也可以没有。比如上例中，第二个 Item 后面的 Index 为 2，表达的是当前 slave Index 为 2 的 sync 的 PDO entry 项目。Item 后面紧跟 Index 的设置，主要是为了唯一区别配置文件中相同类型的 Section。Index 的取值范围：大于等于 0 即可，配置文件的解析软件会根据一个特定的 Section 结构采用对应的解析方法来进行解析的。

3）.Parameter 行

下图为一个 Parameter 行范例：

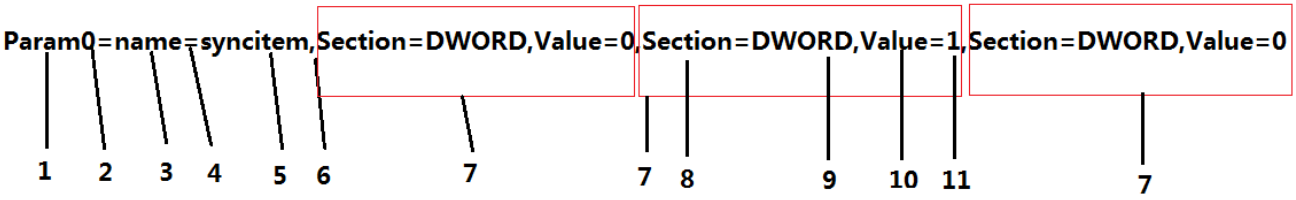

图 2-2　Parameter 行结构图

标号说明：

1. Parameter 行起始字串 Param
2. 当前 Parameter 的 Index
3. Name 标识
4. 连续 tag 间的分隔符
5. 当前 Parameter 名称
6. Parameter 设置项目间的分隔符
7. 一个特定参数段，主要包含参数类型和参数数值
8. 参数类型标识
9. 参数类型（DWORD，HEX，FLOAT，STRING）
10. 参数数值标识

11. 参数数值

一个 Parameter 行主要也就是分为两个部分：（1）行前导，由以上结构图中的1,2,3,4,5 组成，行前导只用于表示单个 Parameter 的名称和所处 Section 的序号（紧跟 Param 后的即为序号），其中的 name 和具体名称都只是便于阅读，没有实际的参数意义；（2）若干参数段，即为图 2-2 中的 7 表达的内容，其中 7 为一个参数段，主要包含8,9,10,11 四个部分，每一部分的含义上面有详细说明，参数段与参数段之间用“,”分割。目前一个 Parameter 行可以跟 10 个参数段，一行中，每一个参数段的参数类型都可以不同，但都只能是以上四种参数类型的任意一种。四种参数类型：DWORD，32 位整数；HEX，16 进制整数；FLOAT，浮点型数；STRING，字符串，最长支持 256 字节。

3.EtherCAT 从站描述文件结构（SII 文件）

基于上一章节的配置文件元结构介绍，我们再来看从站描述文件 SII 就很好理解了，每一个从站都有以下 7 个 Section 组成，分别是：slave_info，slave_sync，slave_sync_PDO，slave_sync_PDO_entry，slave_dc，slave_iomap，slave_startsdo。下面分别介绍每一个 Section 的具体结构：

1）.slave_info

该 Section 的内容为对应 slave 的基本配置信息，以及用于解析以下 6 个 Section 所用到的索引信息。目前该 Section 包含 6 条参数，分别是：

参数类型	名称	数值
STRING	Slavename	厂商定义（来自 xml）
HEX(或 DWORD)	Vid(Vendor ID)	厂商定义（来自 xml）
HEX(或 DWORD)	Pcode(Product Code)	厂商定义（来自 xml）
DWORD	nsync(sync number)	厂商定义（来自 xml）
DWORD	nmap(IO Map number)	根据实际需要设定
DWORD	nsdo(start sdo number)	根据实际需要设定

表 3-1　slave_info 段项目定义

一般情况下，6 个 Parameter 项目中，前面 4 个基本是固定的，后面两个可根据实际情况通过配置工具进行增减。需要注意的是，后面两个参数分别对应第 6,7 个 Section 参数的条目，所以如果手动修改请一定保证索引值和相应 Section 参数条目数对应，否则可能会引起解析软件出错。

2）.slave_sync

该 Section 中有 slave_info 中定义的 nsync 个项目，其中每一条目如以下结构：

Param0=name=syncitem,Section=DWORD,Value=0,Section=DWORD,Value=1,Section=DWORD,Value=0

Parameter 的前导段没有特别含义，syncitem 用于指明该条目配置的是一个 sync 项目，后面紧跟三个参数段，含义分别如下：

参数类型	含义	数值
DWORD	Sync Index	厂商定义（来自 xml）
DWORD	Sync Direction，（1，output；2，input）	厂商定义（来自 xml）
DWORD	该 Sync 所管理的 pdo 数目	厂商定义（来自 xml）

表 3-2　slave_info 段项目定义

此处关于从站 sync 的配置值完全来自于 EtherCAT 从站厂商提供的 xml 文件，当然也必须严格与厂商提供的 xml 对应项相同，否则会出现运行时从站无法配置成功的问题。

3）.slave_sync_PDO

该 Section 的内容对应上一个 Section 的其中某一个 syncitem 条目。因为在一个从站的配置中，可能会有多个 PDO，为了在文件中唯一区别一个 PDO，我们需要在 Section 中“slave_sync_PDO”的“sync”后紧跟一个 index，该 index 为此组 PDO 所对应的 sync 的 index，因此即可以唯一区分一个 PDO，有了这个 index，也可以知道对应 PDO 的方向（dir）。

此 Section 的一个条目如下：

Param0=name=pdoitem,Section=HEX,Value=0x1600,Section=DWORD,Value=2,Section=DWORD,Value=0,Section=DWORD,Value=1

忽略前导段，含义如下：

参数类型	含义	数值
HEX	PDO index	来自 xml 定义
DWORD	PDO 对应 entry 数目	来自 xml 定义
DWORD	PDO 对应 entry 在 entry List 中的 offset	顺序排列产生 offset
DWORD	PDO 被配置与否	用户设置

表 3-3 slave_sync_PDO 段项目定义

这一部分对标准 xml 文件做了一些扩展：在 sii 文件中，包含 xml 定义的所有 PDO，同时顺序记录每一个 PDO 的 entry 总数，以及在所有 entry 组成的列表中的偏移，都在配置文件中体现出来。Entry 列表在下一节就会介绍。且每一个 PDO 通过一个标志，用以标识被配置与否，也即是表示并不是从站 xml 中所有的 PDO 都需要被配置，用户可以根据自己需要选择配置哪些 PDO。

4）.slave_sync_PDO_entry

这一节表述的是 entry 列表，这个 entry 列表对应的是：某一个 index 的 sync 下的所有 PDO 的所有 entry 的顺序集合组成的列表。为了唯一表达这一个 Section，在 Section 的 Item 中的 “sync”后面跟有对应的 index 数值，与上节相同。

此 Section 的一个条目如下：

Param0=name=entryitem,Section=HEX,Value=0x6040,Section=DWORD,Value=0,Section=DWORD,Value=16,Section=DWORD,Value=1

忽略前导段，含义如下：

参数类型	含义	数值
HEX	PDO entry index	来自 xml 定义
DWORD	PDO entry subindex	来自 xml 定义

DWORD	PDO entry bitlen	来自 xml 定义
DWORD	PDO entry 被配置与否	用户设置

表 3-4 slave_sync_PDO_entry 段项目定义

同样，此处的定义也是扩展了标准 xml，前三个参数与标准 xml 相同，第四个参数设置了用户选配项。对应某一个 index 的 sync 下所有的 PDO entry 都会集中在这一个列表中。

5）.slave_dc

此段用于配置设备的 Distributed Clock。一般标准支持 DC 的从站，如果选用的是倍福的 ET1100 芯片，是支持配置两个 syncsignal 的，这里我们仅支持第一个 syncsignal。如果设备不支持 DC，那么此段所有的参数可置零。

此段有四个配置条目，每一个条目的配置含义如下：

类型	含义	数值
DWORD	DC 的 Enable 与否	根据需要设定
HEX	assign_activate 数值	来自 xml 定义
DWORD	Cycle time 数值	来自 xml 定义
DWORD	Shifttime time 数值	来自 xml 定义

表 3-5 slave_dc 段项目定义

6）.slave_iomap

此段是该从站设备所有需要被映射到应用层使用的 PDO entry 的列表。由于某一个从站的 PDO entry 总数是比较多的，但在实际使用是却并不一定需要全部映射，因此采用此段的配置，可以使得 PDO entry 根据需要映射，这样可以有效的节省总线带宽。

根据 sync index 的先后顺序排列所有需要映射的 PDO entry 在这个列表，其中一个 PDO entry 映射条目如下：

Param0=name=mapitem,Section=HEX,Value=0x6040,Section=DWORD,Value=0,Section=DWORD,Value=16

参数的具体含义如下：

类型	含义	数值
HEX	PDO entry 的 Index	来自 xml 定义
DWORD	PDO entry 的 Subindex	来自 xml 定义
DWORD	PDO entry 的 bitlen	来自 xml 定义

表 3-6 slave_iomap 段项目定义

通过配置软件，此段的条目总数会发生变化，对应 slave_info 段的 nmap 参数也会相应变化。

7）.slave_startsdo

有一些支持 COE 的从站设备，在系统上电完成所有配置之后，会有一些参数初始化序列，这个序列一般采用 SDO 的下载方式完成。slave_startsdo 这个 Section 就是用于描述某一个从站启动时 SDO 的下载序列的。当然，大部分从站不需要这个启动下载序列，用户只需要将此段置空即可。

其中一个 SDO 下载条目如下：

Param0=name=startitem,Section=HEX,Value=0x6040,Section=DWORD,Value=0,Section=DWORD,Value=2,Section=DWORD,Value=16

每一个参数的含义如下：

类型	含义	数值
HEX	SDO Index	来自 xml
DWORD	SDO subindex	来自 xml
DWORD	SDO 的 ByteLen（字节长度）	来自 xml
DWORD	SDO Value	根据需要设定

表 3-7 slave_startsdo 段项目定义

通过配置软件，此段的条目总数会发生变化，对应 slave_info 段的 nsdo 参数也会相应变化。

4.EtherCAT 总线配置文件结构（ENI 文件）

ENI 文件的元结构与 SII 文件完全相同，从内容上来讲只是比 SII 文件多若干个 Section，以及采用一种特殊的方式包含一个从站的 SII 信息。

下面分别来介绍 ENI 相比 SII 多出的几个 Section：

1）.CommonInfo

CommonInfo 这个 Section 目前只有一个条目的信息，即版本号。所以只有一条参数：

Param0=name=version,Section=FLOAT,Value=0.01

2）.MasterInfo

MasterInfo 这个 Section 主要包含总线的一些基本设置，有以下参数设置：

类型	含义	数值
DWORD	NIC Index（支持多个网卡，这里选定网卡号）	0~n
DWORD	Debug Level（调试信息输出等级，等级越高输出信息越多）	0~5
DWORD	IO Update Frequency（总线基本的 IO 刷新频率）	1~n （ms），一般不需要设定。

表 4-1　MasterInfo 段项目定义

3）.SlaveIndex

为了提高解析软件的解析速度，这里设置了一个从站目录的 Section，用以告诉解析软件，系统目前挂载有多少个从站。当然解析软件也可以遍历下面所有的从站之后，一样可以知道有多少个从站挂载，但这样的话，解析软件会耗费更多时间在解析上。所以为了减少解析时间，设置了这个 Section，目前只有一个条目：

Param0=name=slavenum,Section=DWORD,Value=8　　//8 个从站挂载

4）.包含 SII 信息的特殊 Section

为了便于解析软件和系统开发人员在 ENI 文件中辨识一个从站的 SII 信息，每一个从站都有一个从站起始 Section 和从站结束 Section：

[SlaveStart]

[SlaveEnd]

这两个 Section 用以显式的表明：这中间所有的设置信息为一个完整的从站 SII 信息。在 SlaveStart 这个 Section，我们增加了若干个条目，用以表达，该从站挂载到总线上时的几个特定配置参数：

类型	含义	数值
DWORD	Position （从站在总线上的位置）	0~n（根据实际拓扑连接设置）
DWORD	Active（该从站使能与否）	0，关闭；1，打开
DWORD	Slaveflag（保留参数）	0
DWORD	Ctrlmode（控制模式）	0，通用模式；1，专业模式
DWORD	type	0，I/O Slave；1，Motion Slave

这里主要对第四个参数做一个说明：一般场合，控制精度要求不是很高，比如 500us 以内的刷新周期，我们都可以选择通用模式；在有一些场合，比如高精度加工，或者高精度机器人部件控制时，刷新周期要求比较高，达到 250us 或者更高，这里就需要设置为专业模式。设置为专业模式，也需要系统硬件的支持，即系统硬件中有专门的 MCU 用以计算高精度控制伺服驱动器。目前版本 ctrlmode 可为默认值。具体可咨询 GoogolTech 的技术支持。

第五个参数，Type：我们将 EtherCAT 的从设备分为 Motion Slave 和 Non-motion Slave，如果是 Motion Slave，则系统默认会采用 GoogolTech 的 Motion 库，而 I/O Slave(即 Non-Motion Slave)则采用通用 EtherCAT Slave 的处理方法，用户在 AP 层自行控制。

5.EtherCAT 总线配置工具 EthercatConfig 简介

为了便于开发人员修改 ENI 文件的各种设置，以及增删挂载到总线上的从站，我们开发了一种可视化配置工具软件 EthercatConfig，如下：

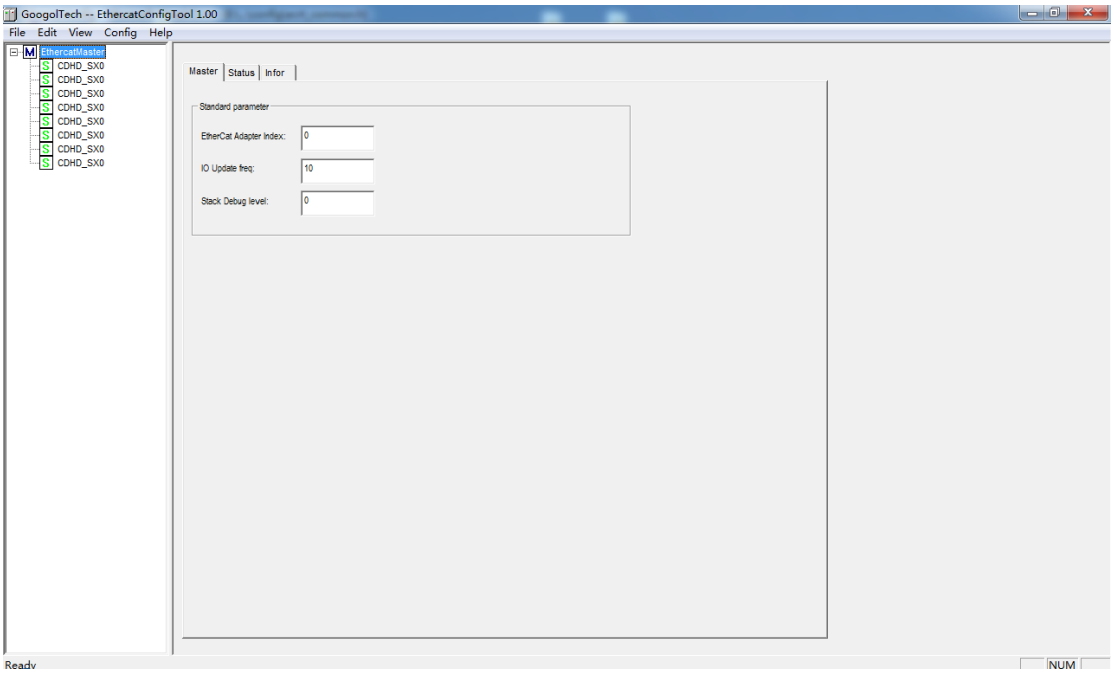

图 5-1 EthercatConfig 配置工具软件

在左侧的树形目录中，显示的是当前总线上连接的所有从站设备，点击任何一个从站设备，右侧的工作空间可以进行该从站的所有 SII 信息配置。

右键点击树形目录条目 EthercatMaster，可以弹出添加设备对话框，在此对话框中可以添加当前 GoogolTech 已经支持的所有从站设备，点击“Add Device”即可加入到总线。

右键点击树形目录的任何一个从站设备，可以弹出删除对话框，点击“OK”，即可删除该设备。

当做完 ENI 的修改后，点击工具条中的“Config”下拉条目“Save Config”，即可把当前所有的修改保存成新的 ENI 文件，拷贝此 ENI 文件到运行时系统，即可被协议栈使用。