基于自动测试系统的测试数据格式标准化研究*

韩 东 郭士瑞 高 剑 李 杰
(北京自动测试技术研究所集成电路测试技术北京市重点实验室 北京 100088)

摘 要：为克服因不同自动测试系统生成测试数据不统一，所造成测试效率低下和成本浪费的问题。根据 ATE 统一测试数据标准格式 STDF 文件的规范，分析其内部各个模块的处理方法，使用 LabWindows/CVI 软件环境，设计了一种转换程序，通过算法完成了二进制文件 STDF 与文本文件的双向转换，通过对比转换生成的文本和标准 ATD 文本，验证了程序转换结果的正确性，最后将转换程序植入国产自动测试系统 BC3192V50，从而实现测试数据标准化。测试实验结果表明，该算法能够高效的实现双向转换，规范了测试数据的结构，提升了测试数据的可分析性。

关键词：自动测试系统；二进制文件；测试数据标准化
中图分类号：TN302 **文献标识码**：A **国家标准学科分类代码**：510.30

Standardization research of STDF based on the automatic test equipment

Han Dong Guo Shirui Gao Jian Li Jie
(Beijing Key Laboratory of Integrated Circuit Testing Technology, Beijing Institute of Automatic Test Technology, Beijing 100088, China)

Abstract: Because of the test data is not unified, the test efficiency of the ATE is low and the cost is wasted. In order to overcome this problem, we analyzed the processing methods of each STDF file module and designed a conversion program using LabWindows/CVI software environment for Specification of STDF files, which is the test data standard format for ATE. By this algorithm, the bidirectional conversion between binary file STDF and text file is completed. By comparing the generated text and standard ATD text, the correctness of the program transformation is verified. In the end, the conversion program is embedded in the domestic BC3192V50 ATE, it realized the uniform of the test data. The result of the test shows that this algorithm can efficiently achieve bidirectional conversion, the structure of the test data is regulated and the analysis ability of the test data is improved.

Keywords: ATE; binary file; STDF

1 引 言

随着全球半导体测试行业的发展，统一测试环境的创造越发成为了测试厂商追求的目标。统一的测试数据格式，将有利于中心数据库数据的规范化组织；有利于使用数据分析软件对测试程序进行调试，省去分析数据有效性问题的时间；有利于忽略测试机和数据库之间的不同环境因素，为普适性、可移植的数据汇总和分析软件创造条件[1]。

Teradyne 公司提出了一种简单、富有弹性、便于移植的数据格式——STDF[2]。它已成为自动测试设备领域以数字形式保存测量数据最为普遍的应用格式，也是几乎可以应用于任何类型测试机的通用文件格式。但是 STDF 文件是一种二进制文件，不能通过打开的方式，直接看到里面的数据。

目前，传统的测试数据研究，更多是停留在理论层面，专注于分析 STDF 文件本身，鲜有提到如何实现 STDF 与文本的双向转换机制。本文以 2007 年第四版 STDF 文件规范为标准，首先研究 STDF 文件的结构及其单元组成、文件的模块划分；其次研究 STDF 文件数据类型的基本含义和格式，各个模块字段的处理方法；然后是编写 STDF 文件和文本文件的高效双向转换程序；最后，基于 BC3192V50 测试系统，将转换程序与测试软件整合。项目意义在于，通过对 STDF 文件相关格式的研究，提出了一种高效的转换算法，在自主产权的国产测试系统上，实现了测试数据结果的 STDF 文件标准化转换。

2 STDF 文件介绍

2.1 STDF 文件特点

在半导体行业发展的过程中，Teradyne 公司为了很好地解决数据管理和记录问题，定义并开发了一种简单、方

收稿日期：2016-12
* 基金项目：北京市自然科学基金委员会－北京市科学技术研究院联合资助项目(L150009)

便、灵活的数据格式。这种数据格式可以被高效率转换，并且可定义为标准测试数据格式(standard test data format，STDF)，目的是让测试机厂商与用户能够在 UNIX、Windows 等不同平台的测试机上，能够统一使用该文件格式进行数据统计与分析。这种灵活、标准的数据测试格式，可以让单个数据模式的程序被大多数半导体厂商以及不同类型的测试机所接受，同时这种格式的文件也很容易从中央数据库搜索引擎导出来做数据分析，并可支持无效和数据丢失数据选项。虽然 STDF 为非 IEEE 标准数据格式，但由于以上优点，世界上绝大部份测试机生产商已经采用了这种标准进行相关的研发[3]。

STDF 文件实际上不是通过定义数据库结构来满足中央数据库搜索与测试机的要求，而是设置一系列的数据记录类，而这些记录类型都是一些特定的数据索引，可以被任何一种分析软件用作普通的数据分析。

2.2　STDF 文件结构顺序

STDF 文件，由多个数据模块构成，如：FAR、ATR、MIR 等，每个模块都是不同的数据类，表示不同的测试信息，但第一个模块必须是 FAR，然后是一个或多个 ATR(ATR 是可选的)，FAR 后面或 ATR 后面紧跟着 MIR，MIR 后面是 RDR(RDR 是可选的)，MIR 后面或 RDR 后面是一个或多个 SDR。每一种数据类型所包含的数据信息是由 STDF 文件规范预先定义的，比如 FAR 的数据内容由 CPU_TYPE(1 Byte)和 STDF_VER(1 Byte)组成。STDF 文件根据需要引用的模块不同，有着最简引用规则[4]，如表 1(带 s 表示该模块可以存在多个)所示。

表 1　STDF 文件最简顺序表

FAR - MIR
FAR - MIR
FAR - ATRs - MIR
FAR - MIR - RDR
FAR - ATRs - MIR - RDR
FAR - MIR - SDRs
FAR - ATRs - MIR - SDRs
FAR - MIR - RDR - SDRs
FAR - ATRs - MIR - RDR - SDRs

3　STDF 文件模块分析

3.1　STDF 文件模块划分

STDF 文件 V4 版本共计 25 个模块，不同的模块被称为不同的记录类型，这些模块按照一定的顺序，共同组合构成了完整的 STDF 文件。其基本结构如图 1 所示：

简单地说，生产测试时，测试设备的信息记录到 MIR。MRR 中，测试 Site 的信息记录在 SDR、PCR 等模块中，被测芯片的参数信息存放到 PIR、PRR、PTR 模块，功能信息

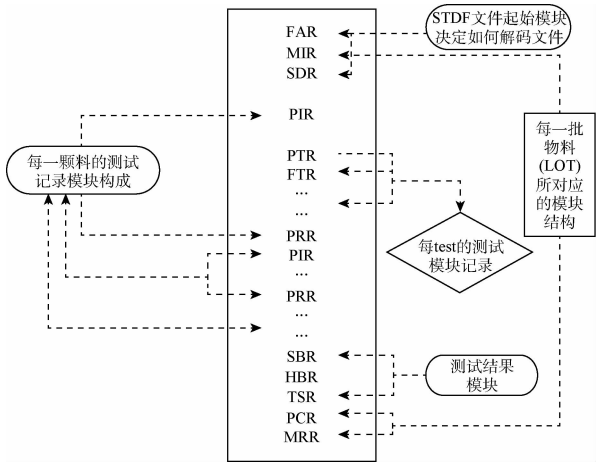

图 1　STDF 文件基本结构

存放到 FTR 模块，这些记录同文件头结合最终形成一个完整的 STDF 文件。图 1 给出的结构图是最为基础性的文件结构，包含了 STDF 文件的基本模块。STDF 文件强大的地方在于，对它的分析只需要判断出是属于哪个模块，就可以进行数据的收集与统计，不用判断其信息是否完整。为了准确识别不同的模块，STDF 文件规定了每个模块的字节代码，只有正确的分析模块字段，才能进行下一步的数据处理[5]。

3.2　STDF 文件字段分析

STDF 文件的每个模块都是由以下 3 个部分组成的：记录标题、必选的数据记录和可选的数据记录。

3.2.1　记录标题

记录标题(如图 2)由以下 3 个部分组成：数据长度(REC_LEN(占 2 Byte))、记录类型(REC_TYP(占 1 Byte))、子类型(REC_SUB(占 1 Byte))。

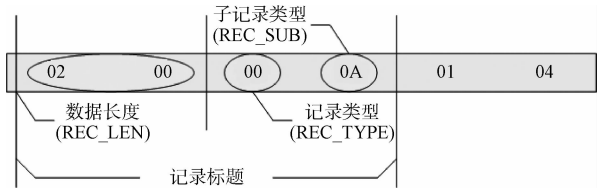

图 2　记录标题说明

数据长度取决于模块长度，再减去 4 Byte(即减去记录标题长度)，记录类型和子类型共同决定了模块的识别。STDF 文件这种定义的优势体现在：在提供了各个模块的识别类型码后，可以让数据分析工具通过组的确定来简化对各个模块的识别过程。因此，进行双向转换之前，必须先确定好整个 STDF 文件的模块识别方式，才能对其中的数据进行处理。

3.2.2　必选的数据记录和可选的数据记录

数据记录，作为 STDF 文件的主体部分，也是进行转换需要提取的有效信息部分。STDF 文件中按照记录类型将

信息分为10个大类,其中最后两个大类是为用户开发预留,在其余8个大类里面将信息依据子类型分为若干个小类,构成了STDF的数据信息[6]。这些信息涵盖面广,从测试系统的信息到实际测试得到的数据,都在各个模块中体现了出来。但是,各个模块中的信息又分为必选的和可选的两个部分。必选的数据记录并不是指数据一定要存在,而是指数据不存在的情况下,需要按照STDF文件数据类型定义规范,用无效的字符,例如,空格(20)、FF(65535)、NULL(00)等进行占位。这样,即使各个模块内的信息之间有缺省的部分,也不会对STDF文件的完整性造成破坏。这也是STDF文件的另一种优势,即不要求所有定义的内容都必须记录,这样可以缩小文件大小,节约存储的空间,并减少产线上的生产时间。

接下来以MRR为例对数据记录内容和类型进行分析,如表2所示。

表2 MRR内容

模块内容	数据类型	功能描述	缺省标志
REC_LEN	U*2	模块长度	
REC_TYP	U*1	模块类型	
REC_SUB	U*1	模块子类型	
FINISH_T	U*4	测试结束时间	
DISP_COD	C*1	晶片顺序码	space
UDR_DESC	C*n	测试文件提供人员	length byte=0
EXC_DESC	C*n	测试执行人员	length byte=0

1)数据记录分析

与记录标题的固定类型不同,各个模块的数据内容都不相同,是由文件提供商和用户共同明确地对每一项内容进行定义。图3是MRR模块的数据记录示例。

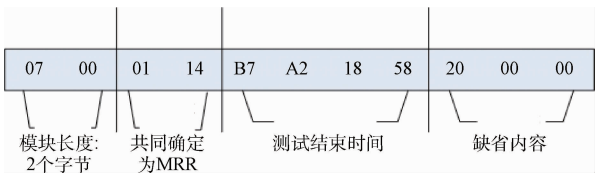

图3 MRR模块

图3中的07 00 01 04是记录标题,后面的B7 A2 18 58 20 00 00是数据记录。07 00表示模块长度为7 Byte,01和14可以通过查STDF模块类型识别表,确定为MRR模块,后面的B7 A2 18 58是占位4 Byte的无符号整数,表示内容是结束时间秒数。秒数加上格林威治时间才是真正的结束时间。

2)数据类型定义

虽然无论从内容还是名称,MRR模块和其他模块之间都会有所区别,但是各个模块的记录标题的识别是一样的,这是因为作为表达因素的数据类型是一样的。比如图3中的长度类型是U*2,代表的就是两个字节的无符号整数占位,相当于C语言中的Unsigned short类型。STDF具有完整而简洁的数据类型定义,如表3列出的是其中与C语言不同的数据类型定义:

表3 STDF文件重要数据类型

数据类型	对该数据类型的描述	对应C类型
C*f	可变长度字符串,长度存储于其他字段	Char[]
C*n	可变长度字符串,其长度第一个字节标识后面字节数	Char[]
B*6	固定长度比特编码字符串,占位6 Byte	Char[6]
B*n	可变长度的比特编码字符串,第1个字节标识后面字节数(<255),即数据项由第2个字节开始	Char[]
V*n	可变数据类型字段,第1字节标识数据类型	
D*n	可变长度的长字节编码字符串,前2个字节标识后面字节数(<65535),即数据项由第3个字节开始	Char[]
N*1	无符号整型,占位半个字节,即第一项位于低4位,第二项位于高4位;基于文件中仅能写入完整字节,则奇数项时,以0000占位最后的高4位	Char
KxTYPE	指定类型数组,K(数组元素数目)在记录之前的字段中加以定义,比如KxU*2	TYPE[]

参照表3中的数据类型,分析图3中MRR模块内容最后3项内容,可以从表2中看出,测试结束时间之后还有3项内容,并且缺省条件分别为空格和两项0。第1个20转化为二进制后为32,即为缺省内容为空格。第2项和第3项内容0表示C*n的缺省值。这个值的具体含义是长度为0,内容为空。内容不为空的情况如图4所示。

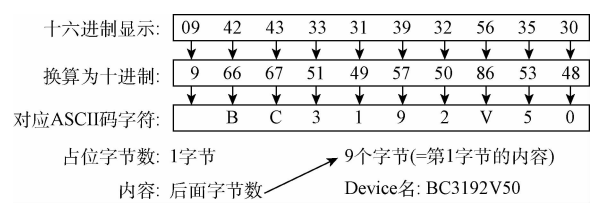

图4 C*n数据类型分析

图4是MIR模块中第10个信息:PART_TYP内容,它代表的是测试部分类型名称,也就是设备名。它的字节

数是由 1+(*n*+1)构成,第一字节标识后面字节的长度。

4　STDF 双向转换机制分析和程序实现

4.1　STDF 文件转换机制

STDF 双向转换机制的核心是如何建立一个规范的文本,文本的规范根据实际需求而定。本文依据 STDF 的特性,创建的文本规范有以下特点:首先文本里面存放的内容按 STDF 各个模块顺序转换后得到,各个模块内容应包括模块起始标志(存放模块名),模块内容(每个信息块按行依次存放,字符':'前存放各项信息名,':'后存放信息具体内容,'\n'识别内容结束、行结束),以及模块结束标志。一个规范的文本,可以降低程序的繁琐程度,提升转化的效率和速度,实现双向转换。

4.2　STDF 文件转换为文本文件

开发 STDF 文件转化为文本文件的程序,是为了能够直观的对 STDF 文件所存储的信息进行显示和分析。文本的格式在 4.1 中已经具体说明,以 MRR 为例,图 5 显示的是定义的规范文本。"MRR:"是模块起始标志,"END_MRR:"是模块结束标志,中间为 4 个内容项,表 2 给出了各项内容的意义。

```
5531 MRR:
5532     Date and time last part tested:2016年11月7日15时31分13秒
5533     Lot disposition code:
5534     Lot description supplied by user:
5535     Lot description supplied by exec:
5536 END_MRR
5537
```

图 5　MRR 文本格式

转换基于 LabWindows/CVI[7-8] 环境,采用 C 语言,编写程序。主函数 main()中采用 fread()函数,从 STDF 文件头开始,每次先读取两个字节的长度信息(REC_LEN),如果为 0,说明 STDF 文件为空;如果不为 0,再分两次取 1 个字节的类型(REC_TYPE)和子类型信息(REC_SUB),识别模块类型,采用 switch()函数的嵌套,调用各个模块对应的子函数,最后在子函数中按照该模块内容定义及其数据类型依次读取,处理,再按照规定格式输出到文本文件。程序结构见图 6。

接下来用 MRR(见表 2)模块作为例子,进行分析。MRR 模块始终位于 STDF 文件的最后,其功能是测试结果记录,图 7 是用 UltraEdit 软件打开的 STDF 二进制文件内容:

图 7 中选中的部分全部属于 MRR 模块,此时文件指针已经位于 MRR 首部,具体内容的处理已在 3.2.2 中作过分析,这里不再赘述。

程序的验证,是分模块进行验证,将 STDF 文件内容转换为文本后,再利用工具将 STDF 文件转换为 ATD 文件,它能够查看 STDF 模块中的各项内容,对应各个模块的各项信息,依次比对验证。

4.3　文本文件转换为 STDF 文件

文本转换为 STDF 文件,是实现 BC3192V50 输出数据

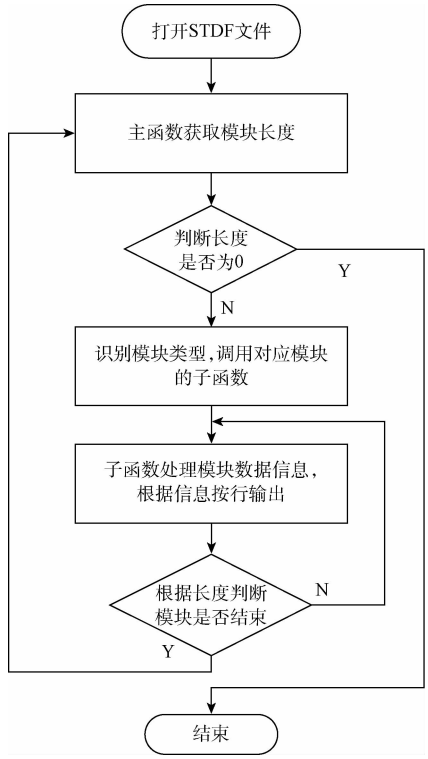

图 6　STDF 转为文本的程序流程

```
0001f1d0h: 0F 00 00 00 00 00 00 00 00 00 00 0E 00 00 00 ;
0001f1e0h: 00 00 00 00 16 00 01 1E 00 02 0F 00 00 00 00 00 ;
0001f1f0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 16 00 ;
0001f200h: 01 1E 00 03 0F 00 00 00 00 00 00 00 00 00 00 ;
0001f210h: 00 00 00 00 00 00 00 00 07 00 01 14 B7 A2 18 58 ;
0001f220h: 20 00 00                                       ;
```

图 7　MRR 模块二进制内容

标准化的关键步骤。难点一方面在于如何精确的识别出模块,这就是本文在设定文本格式时,加上起始和终止两个标志位的原因。通过查找两个标志位能够精确判定模块的开始与结束位置。另一方面在于需要精确计算出各个模块所占的长度,用图 5 作为例子,可以看出最后 3 项内容缺省,但是模块的长度需要在模块头定义下来,所以在模块读取的过程中,需要对模块的长度变量进行实时的修改,最后按照 STDF 文件模块规定的数据类型进行输出。

文本转化为 STDF 文件的程序,有两个重点:1)对于其中模块长度的处理,有两种方式能够实现。(1)采用在最后集体输出数据信息的方式,这样长度信息在最后输出,优点是输出集中,方便后期维护;缺点是占用过多的变量,因为所有变量都需要在最后进行输出,不能在中间释放。(2)采用一边处理文本,一边输出的方式,可以省去过多的中间变量,但是需要最后将文件指针移到长度位置修改,这样会导致程序处理时间过长。本文结合实验调试,采用第一种方式进行处理。

2)处理数据类型,接下来对重点数据类型进行分析。U*1,U*2,U*4(无符号整型),I*1,I*2,I*4(有符号

整型)的处理类似,都是将文本里的内容,存放到字符串中,提取出其中有效的部分处理输出;C * n 的处理已在图 4 中进行过分析,D * n 与其类似;B * n 是按位处理的类型,需要将十进制转换为二进制比特位,然后根据 STDF 文件规范去处理文本;V * n 需要先参考使用类型,再往 STDF 文件里写入具体内容;N * 1 注意的是,缺省时,在 STDF 文件里没有占位;KxTYPE 需要先计算出 K 的值,再去按照规定的类型进行处理[9]。

由于程序结构与图 6 相同,只是打开的文件为文本,所以这里不再对程序结构进行分析,以本文讲解最多的 MRR 子函数进行剖析,图 8 为 MRR 子函数处理程序。

```
void MRR(void)
{
      int i,l,h;
      unsigned long finish_t;
      unsigned short rec_len;
      char rec_typ,rec_sub;
      char disp_cod,len1,len2;
      char ch1[50],ch2[50];
      rec_len=0;                    //模块长度初始化
      rec_typ=1;                    //模块类型初始化
      rec_sub=20;                   //模块子类型初始化
      trans_time();                 //调用格林时间转换函数
      finish_t=shijian;             //结束时间初始化
      rec_len=rec_len+4;            //总长度叠加上时间信息所占长度
      fgets(str,260,fp_ywj);        //文件按行进行处理
      disp_cod=str[FindPattern(str,0,-1,":",0,1)+1];//寻找信息位置并赋值
      rec_len++;
      fgets(str,260,fp_ywj);              //文件处理下一行
      l=FindPattern(str,0,-1,":",0,0);    //寻找信息USR_DESC起始位置
      h=FindPattern(str,0,-1,"\n",0,0);   //寻找信息USR_DESC终点位置
      len1=(char)(h-l-1);                 //计算出信息长度
      rec_len=rec_len+len1+1;             //总长度加上信息长度和信息内容长度
      for(i=0;i<len1;i++)
            ch1[i]=str[l+1+i];            //把信息放进字符串
      ch1[len1]='\0';                     //字符串末尾加上结束标志
      fgets(str,260,fp_ywj);
      l=FindPattern(str,0,-1,":",0,0);    //寻找信息EXC_DESC起始位置
      h=FindPattern(str,0,-1,"\n",0,0);   //寻找信息EXC_DESC终点位置
      len2=(char)(h-l-1);                 //计算出信息长度
      rec_len=rec_len+len2+1;
      for(i=0;i<len2;i++)
            ch2[i]=str[l+1+i];
      ch2[len2]='\0';
      fwrite(&rec_len,sizeof(unsigned short),1,fp_copy);  //按序将信息输出
      fwrite(&rec_typ,sizeof(char),1,fp_copy);  //输出记录类型
      fwrite(&rec_sub,sizeof(char),1,fp_copy);  //输出子类型
      fwrite(&finish_t,sizeof(unsigned long),1,fp_copy);//输出时间
      fwrite(&disp_cod,sizeof(char),1,fp_copy); //输出晶片顺序码
      fwrite(&len1,sizeof(char),1,fp_copy);     //C*n输出前先输出长度信息
      for(i=0;i<len1;i++)
            fwrite(&ch1[i],sizeof(char),1,fp_copy);//用字符串输出C*n内容
      fwrite(&len2,sizeof(char),1,fp_copy);
      for(i=0;i<len2;i++)
            fwrite(&ch2[i],sizeof(char),1,fp_copy);
      return ;
}
```

图 8 MRR 模块转换函数

函数的前半部分用到最多的是查找关键字,获取关键字后面有效字段的内容,分别得到了 REC_LEN、REC_TYP、REC_SUB、FINISH_T、DISP_COD、UDR_DESC、EXC_DESC 字段对应的内容,并保存在不同变量中。函数的后半部分将上述内容依次按照 STDF 标准写入二进制文件。

程序的验证分为两点,一是将转换生成的 STDF 文件,与以前的 STDF 文件进行比对,比对分模块进行,由于比对数据量大,只需验证数次各个模块中存放的二进制数是否正确;二是将生成的 STDF 文件转换为文本,同时利用工具软件将其转换为 ATD 文件,以文本比对的方式将二者进行比对。从而借助第三方软件排除了因程序双向转换相互影响而造成的漏洞。

5 BC3192V50 测试系统输出数据标准化实现

5.1 BC3192V50 软件植入 STDF

基于文本转换 STDF 文件程序,对于实现自动测试系统 BC3192V50 数据标准化,只需将文本转 STDF 文件程序植入到 BC3192V50 软件环境中。

整合的难点在于生产时,在什么位置对 STDF 文件进行转换,有两种可行方案。1)采用将测试系统数据输出文本作为规范文本格式,等生产测试结束后,单独的对文本进行处理,生成 STDF 文件。此方案优点是不会占用测试时间,STDF 文件的修改方便;缺点是当测试系统数据输出需要动态地修改或者添加内容时,程序通用性降低,限制了 STDF 程序的使用,有较大的局限性。2)在测试的过程中实时地生成 STDF 文件,随着每个测试项的执行,测试数据往文本输出的同时,也往 STDF 文件里进行输出。这样的优点在于 STDF 程序通用性强,结构清晰,缺点在于依赖测试系统数据输出,影响 STDF 文件内部模块顺序,不利于软件的可修改性[10]。

综合考虑,本文采用的是第二种方案,在测试系统输出文本信息的同时,进行 STDF 文件的输出。根据 STDF 文件各个模块的相关顺序,将 4.2 节所述程序与测试软件进行了整合,实现了 BC3192V50 输出数据的标准化。

5.2 实验验证

实验采用 74F138 数字芯片为样本,测试 Site 为 0、1、2、3,有效 Site 为 0、1,测试次数为 20 次,测试总量为 40,测试时 Site 0 失效 5 个。实验主要验证内容为:信息记录模块 MIR 中的测试时间、测试机类型、生产批号、测试机名、测试设备名、测试程序名、软件版本;Site 信息模块 SDR、PCR;参数测试模块 PIR、PRR、PTR,功能测试模块 FTR 等与实际测试需要相符号的模块内容。图 9 是 BC3192V50 测试软件生产流程主界面,MIR 模块等信息均在图 10 中验证,各 Site 良率信息可以从图 11 中验证。其

图 9 测试系统主界面

他各个模块信息的验证，与之类似，不必一一列举。

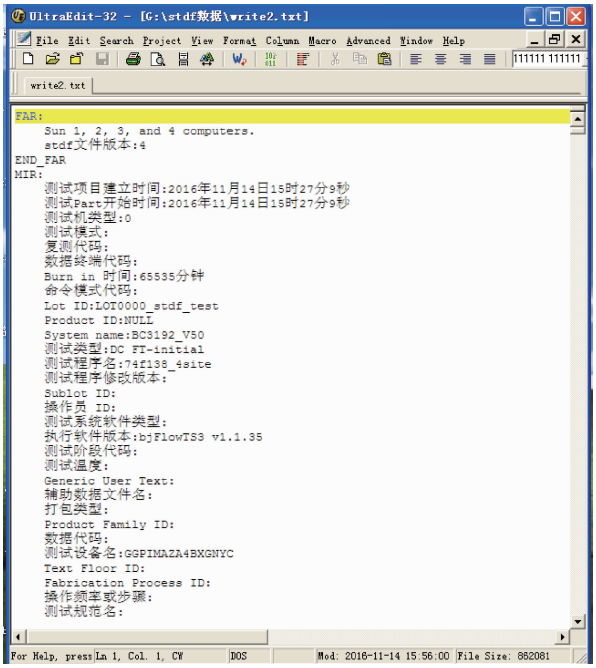

图 10　MIR 模块文本

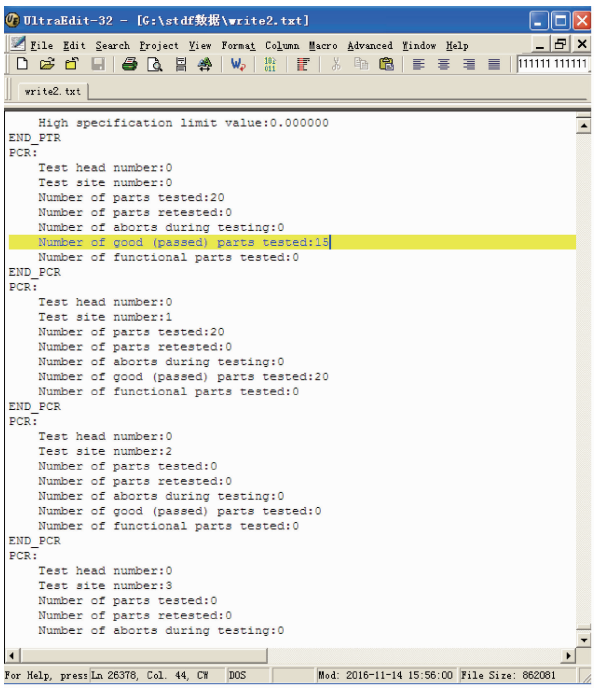

图 11　PCR 模块文本

6　结　　论

本文简单介绍了 STDF 文件的特点、模块信息、字段特点，开发了一种 STDF 文件与文本文件双向转换程序，编写了双向转换的程序实现算法。最后采用此方法，将程序植入 BC3192V50 测试系统，经过对所有模块的一一验证，此转换程序植入 BC3192V50 测试系统后，可以高效的将测试数据转化为 STDF 文件，实现了国产系统 BC3192V50 测试数据标准化。对于 BC3192V50 生成 STDF 文件的效率，STDF 内部模块顺序的合理性，以及随着 BC3192V50 的提升，STDF 文件转换算法的扩展，都有待进一步研究。

参考文献

[1]　袁薇. 标准测试数据格式(STDF)文件的研究[J]. 电子元器件应用，2009，11(4)：70-73.

[2]　Standard Test Data Format(STDF)Specification[S]. America：Teradyne，2007.

[3]　俞生生. 半导体封装测试设备自动化系统的设计与实现[D]. 上海：上海交通大学，2013.

[4]　郑立钧. 利用 JAVA 对 STDF 文件进行分析[J]. 电脑知识与技术：学术交流，2007，1(4)：1017-1018.

[5]　魏宁. ATE 系统的结构分析与数据管理优化[D]. 天津：天津大学，2005.

[6]　陈三定. 通用文件格式转换工具的设计与实现[C]. 中国新闻技术工作者联合会 2008 年学术年会论文集(上)，2008.

[7]　薄志峰. 基于 LabWindows/CVI 的电动舵机自动化测试系统设计[J]. 国外电子测量技术，2015，34(5)：66-69.

[8]　邝继顺，周颖波，蔡烁，等. 一种用于测试数据压缩的改进型 EFDR 编码方法[J]. 电子测量与仪器学报，2015，29(10)：1464-1471.

[9]　任浩琪，林正浩，熊振亚. 基于分治策略的加法器测试向量生成技术[J]. 仪器仪表学报，2016，37(5)：1172-1179.

[10]　王珺，叶卫东. LXI 数据采集器软件测试方法[J]. 电子测量技术，2016，39(4)：140-144.

作者简介

韩东，工学学士，研究实习员，主要研究方向为测试原理、测试系统优化、测试数据标准化等。
E-mail：18687087250@163.com