XDJ-73小型电子计算机

系统程序参考资料

动态调试程序

（ODT）

中国科学院

北京电工研究所五室编

目　　录

一、关于DDT的一般概念

对于一个新程序可以在不同水平上来进行调试以排除错误。在最粗糙的水平上进行改错工作，程序员可以简单地把程序存入内存然后令之运行直到程序出乎予料地仃下为止。之后程序员可以利用控制面板上的开关和键，观察指示灯的显示以试验和找出错误。这种解决方法有两个问题。第一，程序仃下之时错误可能使得所有有关信息包括错误本身发生变化或消失，亦即程序可能由于简单的自我破坏过程而仃下来。第二，程序也可能根本不仃下来，而是连续不断的进行无穷循环，这类循环不易检出。

如果程序员事先规划其改错工作，他可以利用计算机的控制面板在程序开始之先予先放置仃机指令。每次仃机之时，他可以检查各寄存器的内容，必要时予以改变。程序员用原始指令将每个仃机指令替换下来就可以较方便地找到错误的根源。然而错误很少按予期的那样出现。因而这种办法的实际用处是有限的。

此外将二进制的灯光显示变成易于辨认的符号表达式也是很困难的任务。尤其是给一个程序加进一些修补性的补充往往需要大量的开关操作。因此程序员利用控制面板来调试程序往往是得不偿失。

可以设想程序员原来用控制面板所能完成的修改程序错误的任务，转由利用一个专用程序来代替，这样一个专用程序应能满足哪些要求呢？首先它应能使程序员不再利用控制面板上的开关和键就可以完成检查内存内容，改变其内容等修改程序的任务。其次这种专用程序还应易于置换或去掉仃机指令，甚至于自动恢复和执行被仃机指令所替换的那些指令。更重要的是，这类程序能使程序员利用电传键盘和打印机完成符号语言形式的检查和修改任务。这种专用程序应能自动完

成将二进制编码翻译成符号语言的任务。

本机提供的动态调试程序（Dynamic Debugging Technique简称DDT）就是具备上述功能的一种专用程序。借助它可以缩短目的程序的调试时间。程序员将目的程序和DDT均存入计算机内存中，然后通过DDT对目的程序进行各种调试工作，诸如控制目的程序的执行，修改目的程序或其数据，以发现目的程序中的各种潜在错误，并加以修改，以保证目的程序的正确执行。下面首先介绍一下DDT的基本功能和使用DDT的一般问题。

1. 调试目的程序的准备工作

当程序员打算开始对一个新编的目的程序进行调试时，首先应该准备好下列资料：

(1) 目的程序的二进制纸带；

(2) 符号表纸带（为汇编输出的一部份）；

(3) 用符号语言列出的新程序表；

(4) 用符号形式列出的符号表；

(5) DDT本身的二进制纸带。

待调试的新程序和DDT程序均用二进制引导程序送入内存。这时在计算机中有：

(1) DDT程序，它占有5240到7600内存单元，计1248个内存单元。

(2) 目的程序所占用的内存不能复盖DDT程序及其符号表所占用的内存。

(3) 符号表占用从5240到5000的内存单元。这个符号表包括主机的访内指令，运算微指令，10个基本输出入指令，以及复合微指令CIA和LAS。这个符号表详见附录二。

因为DDT完成二进制编码和符号语言之间的翻译工作，所以它必须将用户定义的符号表存入内存中。为了输入符号表纸带应该用电传读带机按下列步骤进行。

(1) 将读带机断开，将符号表纸带放好。

(2) 按ALT MODE（或ESCape）键及R键，结果打印出（(R)符号，然后将读带机接通。

(3) 当计算机打机时断开读带机按"继续"键。DDT将打印出符号表所占用的内存单元的最低地址。

用户定义的符号表在内存中存放的位置紧接在固定符号表的后面。用户定义的符号表与其它从控制面板存入的符号一起组成外部符号表。这些符号随时都可清掉而不影响固定符号表。

当内存中有了上述内容之后，就可开始调试工作了。

2. DDT的基本作用

下面举一个将N个整数加起来求其总和的简单例子，来说明DDT的基本作用。程序分主程序和子程序两部分。

```
/整数相加的子程序
INTSUM,  0
         CLA
         TAD I INTSUM   /取得数据
         DCA   N
         DCA   PSUM
LOOP,    TAD   PSUM     /进行计算
         TAD   N
         DCA   PSUM
         ISZ   N        /调整变址
```

```
        JMP    LOOP          /未完
        TAD    PSUM          /已加完将结果放入AC
        ISZ    INTSUM
IEXIT,  JMP I  INTSUM        /返回主程序
N,      0
PSUM,   0
*400
ITEST,  CLA                  /主程序
        JMS    INTSUM
INT,    0                     /自变量
RTN     HLT
$
```

对于这个具体程序，调试步骤如下。首先将整数放在符号地址为INT的内存单元中。这项工作当利用DDT完成时，操作如下。先用电传打出内存地址INT，再打一个斜画，即

INT/

这样程序员就告诉DDT，他打算检查该内存单元的内容。于是DDT在斜画后面以打印方式进行回答，回答的就是该内存单元存放的内容。在本例中这个内容现在是0，于是结果为：

INT/ 0000

这里下线表示这些字符是DDT回答时打印出来的，以区别于程序员打印出来的字符，但在实际操作时并不打下线。以后各节同。

完成上述操作后，DDT打五个空格之后等待程序员输入新的命令。这时该内存单元处于"打开"状态，这意味着对其内容可以进行修改了。现在假设程序员打算把从1到10，等10，个数累加起来。

这些数必须用八进制数表示，因为DDT不进行十进制运算。又假设程序员从10₈开始加起，于是他打出10并按"回车"键。

INT/ 0000 . 10↵

这样，就将10存入INT单元中，并将之封闭，如果不用前述方式将之重新打开，就将永远保持这个数据。

这时，程序员可以启动目的程序。如果程序工作正常则启动后几乎立即可以得到结果，即AC的内容应是八个数的总和44₈。启动目的程序的操作是：先打出目的程序的启动地址，按ALT MODE键后，打印出左方括弧〔，其功能是表示随后的字符为DDT命令，再打印便目的程序开始执行的命令G，

INTEST 〔 G

于是目的程序开始执行。但立刻发现程序不对，因为AC显示不是44₈。不过这时不知道程序错在那里，为了找出错误，程序员可以进行断点操作，所谓断点操作就是DDT容许程序员在任何地点将目的程序中止执行的功能。这时又重新转回去执行DDT程序。断点操作先打出断点地址再打出断点命令B。例如程序员如果打算在INTSUM+3处将目的程序断开，即数据已取到AC，但还未存入N内存单元中，就按下列形式给出命令

INTSUM+3 〔 B

这时DDT进行下列操作：将断点内存单元中的指令（这里是DCA N）移到DDT指定的一个内存单元暂存。在此指令处另存入一条JMP指令代替原指令，以便目的程序执行到这里时将程序转到DDT去。

为了肯定出现的错误并没有破坏INT内存单元的内容，再将之打开。

```
INT/ 0010
```

确认其内容无误后，再用下述命令启动目的程序。

```
ITEST ( G
```

目的程序立刻到达断点。程序转到DDT。这时在DDT的指挥之下，打印断点地址，右圆括弧，以及保存的AC的内容。

```
INTSUM+0003 ) 0010
```

从打印的结果判断，从主程序向子程序传送信息的过程是正确的。

于是程序员又以同样方式将断点移到子程序的IEXIT处，发现在这里出了问题。他知道问题在于循环部分，于是在LOOP部分用断点观察，进而去检查N中的数据。给出下列命令：

```
LOOP+2 I B
ITEST ( G
LOOP+0002 ) 0010
```

从打印结果看出，在第一次循环之后AC的内容仍等于N的起始值，这是正确的。但是这时N的内容应已改变。如果子程序工作正确，N的内容在第一次循环之后应等于7。但在进一步处理时打开N后发现

```
N / 0011
```

根据这个结果程序员可以判断究竟问题在哪里。原来ISZ指令是将变址器内容加1而不是加I。如果原来在INT中放一个负数就对了。至此问题已经查清，将程序修改过后，再经调试就可结束了。

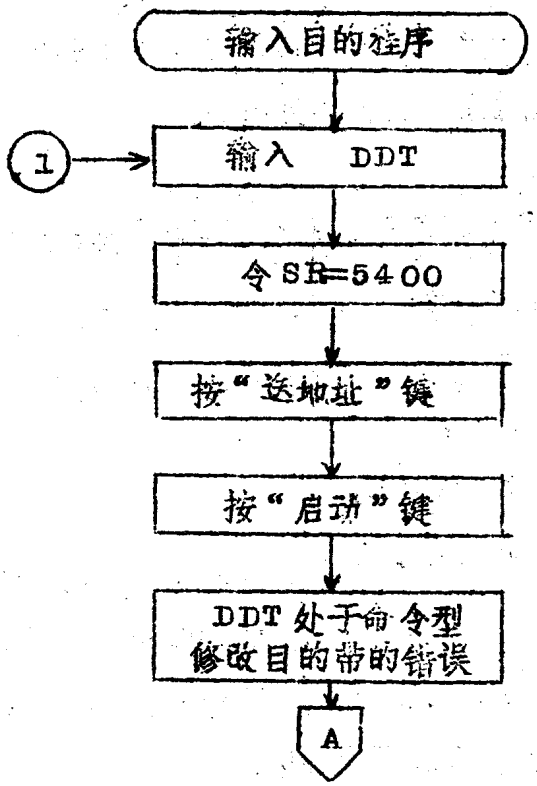
DDT的详细功能将在以后加以介绍。

DDT 的 功 能

1. 存贮要求　DDT的运行部分占有的内存容量从内存单元

5240到7577。固定符号表占有的内存容量从内存单元5237到5004。固定符号表的符号见附录2。外部符号表最多可以存放250个符号，其所用内存容量从内存单元5000到3030。如果留给外部符号表的容量用不完可以为用户使用。每个新定义的符号在外部符号表中占用四个内存单元。在运行时，DDT使用内存单元0004作断点联系用，因此这一单元不能由用户自由使用。程序的起动地址为5400。

2. 输入步骤　当目的带已经存入内存时，再输入DDT程序，两种程序都用二进制引导程序输入。



3. 符号表纸带　汇编穿孔纸带输出的一部分是用户定义的符号表。这个符号表纸带作为DDT的外符号表按下列步骤只能用电传读带机输入。

3.1 将读带机断开，放好纸带。

3.2 按ALT MODE键，然后按R键（印出〔R），将读带机接通。

3.3 当计算机打机时，断开读带机，按"继续"键。DDT将外符号表所占用内存最低地址打印出来。

3.4 如果输入不止一条带，则令SRO＝0，对于每一条带重复3.1～3.3步。

读带将继续到达到带尾或已读入250个符号时为止。如果已达到此最高限值，则除非清掉一些就不能加进更多的符号。但即使在纸带中间就已到达极限值，用户仍可进行程序改错工作，方法是按EOT键，将读带机断开，按下"继续"键。尚未读入的其余符号不包括在符号表中。输入外部符号表纸带的流程图详见附图。

4 定义

符号（Symbol）就是最多可到六个的一串字母和数字，头一个必须是字母。下列是合法符号：

FIMAGE，

K2，

X464PQ，

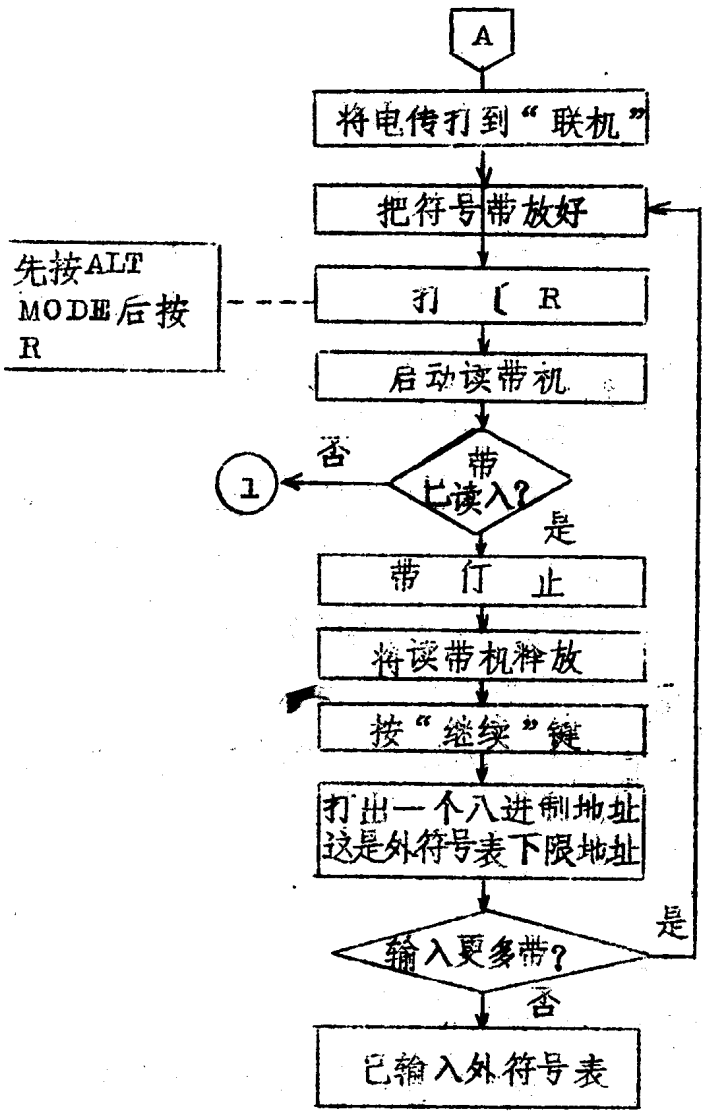
PMLA。

而下列则不能采用：

4WD	不是以字母开始
F2.8	包含非法字符
AN PRG	在符号中间不能插入"间隔"符号
GANDALF	多于6个字母

数字（Number）是四个八进制数串的组合。因此数字的最大值

输入外部符号表纸带流程图



这个操作只用低速读带机

为 7777₈。数 8 和 9 只能在一个符号中当作字符使用。

表达式（ expression ）可以是一个符号，一个数字或一系列符号和数字由下列之一的运算子（ operator ）结合在一起。

+	算术加法
−	算术减法
间隔（ ␣ ）	表示表达式其余部分作为指令的地址来处理

所有其它字符除了用于DDT控制命令的字符以外均为非法的。

如果两个或两个以上的“间隔”符号接连出现，除了第一个之外均被略而不计。因此，

```
TAD     TEM
```

和 `TAD TEM` 是完全相同的。

DDT在CR，LF（回车，换行）的组合中有多余的“CR”时仍能有反应，而在其它情况下多余的CR符号将被忽略。

有下列错误时将使DDT打出一个“?”号，并将从出错点到它前面的“TAB”或“CR”符号之间所有打印出来的信息忽略不计。这些错误是：

(1) 未定义符号，非法符号。

(2) 非法字符。

(3) 未定义的控制命令。

(4) 跨页编址（Cross-page addressing）。

5 型式控制 任何含有符号的表达式叫做符号式表达式，而一个只含有数字的表达式叫做八进制表达式。用户可以按其需要和便利打按任何型式的信息。但DDT的输出则仅仅打印一种型式的表达式，究竟输出那种信息由下列命令来规定。当DDT投入运行之初其输出信息是符号型式的。

↑O 此命令使 DDT 印出的信息是八进制的。键盘输入可以是符号的或八进制的均可。如 LOC = 2642，则

(1) LOC/1263（输入为符号）

(2) 2642/1263（输入为八进制数字）

不论输入是什么型式，输出均为八进制数。

↑S 此命令使 DDT 印出的信息是符号型式的。键盘输入既可是符号也可是八进制数。如果 LOC = 2642，C(LOC) = TAD DATA+4，则

(1) LOC/TAD DATA+0004

(2) 2642/TAD DATA+0004

不论输入是符号还是八进制数，输出都是符号型式。

如果用户想得到他自己打出的或 DDT 打出的符号表达式的八进制值，但又不改变输出的型式，可以使用下列命令。

= 在一个符号表达式之后立即打印此命令，它将使 DDT 印出符号表达式的八进制值。

1. LOC = 2642

2. LOC/TAD DATA+0004 = 1263

上述第2例中输出为符号型式，在应用"="号后重新输出为外内容时仍为符号型式。

6. 检验和修改程序　下面这些命令帮助程序员检验和修改任何内存单元的内容。但要注意不要将 DDT 程序及其符号表本身占用的内存单元打开和修改，DDT 本身没有保护措施，这样做将产生错误工作。

/ 这是一个内存单元检验字符。在表达式之后打印，则 DDT 将把表达式所表示的地址的内容打印出来。例如如果用户打印

LOC/

则 DDT 立刻打印 LOC 的内容然后继之以五个“间隔”符

LOC/TAD DATA+0004

如果需要，用户之后可以更换内存单元的内容如下例。

LOC/TAD DATA+0004　　JMP LOC+10

)(CR) 这个命令将使 DDT 在对打开的内存单元的内容进行所要求的更换之后（如果需要更换的话）将该单元关闭。

LOC/TAD DATA+0004　　JMP LOC +10)

如果打印了多余的“ CR ”，对 DDT 的操作无影响。

换行(LF) 如果在检验和（或者）修改一个内存单元的内容之后，用户希望依次打开下一个内存单元，这时可以打印“换行”键而不是“回车”键。则已打开过的内存单元关闭，而 DDT 打开下一个内存单元，打印其地址，一个反斜画“＼”（此符号表示该内存单元不是用户打开的）然后再打印新内存单元的内容以及另外三个“间隔”符号。例如在检验并互换了 LOC 的内容之后，用户希望检验 LOC+1 的内容，则

LOC/TAD DATA+0004　　JMP LOC+10(LF)

LOC+1＼ DCA DATA

现在 LOC+1 内存单元已被打开。“换行”符号可以在任何时候使用。甚至于如果已经检验过的上一个内存单

元已经关闭或者中间插入了其它操作之后都可使用。例如

LOC/TAD DATA+0004　JMP .+10↓

(0

.+5IB

(LF)

这时 DDT 将仍能使 LOC+1 内存单元打开。断点地址对于DDT 内部的计数无影响，它只跟踪上一个被打开过的内存单元。

↑ 如果用户不是想更换被打开的内存单元的内容，而是想检验其内容所表示的那个地址的内存单元，则使用此命令。例如

LOC/TAD DATA+0004　↑

DATA+4\OPR+337

现在 DATA+4 的内存单元被打开了。

一般这个操作多用于不修改内容的内存单元。如果在经过修改内容之后打此符号，如同下例

LOC/TAD DATA+0004　JMP LOC+10↑

则 DDT 在下一行打印出

DATA+4 \ JMP LOC+0010

即这时不是更换内存单元 LOC 的内容，而是更换内存单元 DATA+4 的内容。

间接地址符号不能被“↑”符号辨认如果 C(LOC) = TAD I DATA+4 则当

LOC/TAD I DATA+4　↑

则“↑”符号仍将DATA+4内存单元打开。

•（句点）这个命令的含义是其值等于上一个被打开的内存单元的地址。其用法有几种。

例1. 核对修改的结果，如

```
LOC/TAD DATA+0004   JMP LOC+10↓
•/JMP LOC +0010
```

例2. 表示刚被打开的内存单元，如

```
LOC/TAD DATA+0004   JMP•+10↓
```

例3. 执行任何一个命令，这个命令以刚被打开过的内存单元的地址为相对起点，如

```
LOC/TAD DATA+0004   JMP•+10↓
•-5(G
```

← 打出这个命令将清掉错误符号。位于←号和前一个“TAB”或“CR”符号之间的信息均被略去，DDT回答一个“TAB”信号。例如

```
                                这是DDT回答的“TAB”符号
                                        ↓
LOC/TAD DATA+0004   JMP LC←     JMP•+10
                    这些符号被取消
这里是TAB符号       这是重打的正确内容
```

7. 跨页编址（Cross-page addressing） 当用户想将一条指令放入一个被打开的内存单元，则这条指令涉及的地址必须和被打开的内存单元的地址在同一内存页。否则就是跨页编址，DDT将回答一个“？”号并将该信息略去。例如：

如 LOC＝2642， XPAG＝2770， 则

LOC/TAD DATA+0004　　DCA XPAG+20↵

?

因为XPAG+20＝3010宅和LOC不在同一内存页，所以DDT回答"?"号。这时LOC内存单元将不作任何修改而关掉。

反之，如下例

如 LOC＝2642， XPEG＝3010 则

LOC/TAD DATA+0004　　DCA XPEG-20↵

将被执行，因为XPEG和LOC虽不在同一内存页，但XPEG-20＝2770则和LOC在同一内存页。

8. 复合微指令和输出入指令的使用

除了CIA和LAS外，复合微指令及输出入指令（电传使用的除外）在固定符号表中不予定义。将这类指令放入打开的内存单元中时，复合微指令不能超过两个记忆符号，且第2个必须是CLA。任何另外的组合将作为错误处理，其信息将被略去。例如下例情况视为错误：

XPAG / CLA　　CLA CMA↵

?

而改做下列写法就是正确的：

XPAG/CLA　　CMA CLA↵

如果复合不包含CLA，程序员可以做两件事情之一。一种是将新的组合符号（其值等于复合码）加以定义存入符号表。例如

CLL RAR　可以定义为CLAR其值为7110。

另一种是程序员可以将表达式表示成包含符号OPR。例如上例的CLL RAR可以写成OPR+110。程序员也可以将输出入指令的组