INFORMIX—4GL技术丛书之三

交互式调试程序

北京希望电脑公司

■ 北京市新闻出版局
■ 准印证号：3320-9032
■ 订　　阅：北京8721信箱资料部
■ 邮　　码：100080
■ 电　　话：2562329
■ 乘　　车：乘332、302和[illegible]
　　汽车到海淀黄[illegible]
■ 办公地点：希望公司大楼101房间

目　录

序 言

本指南说明

INFORMIX—4GL交互式调试程序指南既是对Debugger调试程序的介绍，也是Debugger调试程序命令及特性的综合参考手册。下面将概述本指南的结构，后面做详细讨论。

- 第一章总括了Debugger调试程序的特性，第二章至第七章介绍调试环境及一系列调试对话的典型调试策略。在使用了这些对话之后，将学会适用于4GL应用程序的调试策略。
- 第八章及第九章是参考部分，它们对Debugger调试程序的全部命令及特性进行了详细讨论。

对于INFORMIX—4GL交互式调试程序有二点需附加说明：

- INFORMIX—4GL交互式调试程序通过一个典型的调试对话给出了Debugger调试命令及特性的概括介绍。
- INFORMIX-4GL快速参考卡归纳了INFORMIX-4GL和Debugger调试命令的语法。

（1）本指南的结构

为了满足Debugger调试程序新老用户的不同需要，本指南分两部分：一部分是含有七章的指导部分，另一部分是含有两章的参考部分。

指导部分

- 第一章，“序言”，总括了Debugger调试程序的特性。
- 第二章，“Debugger调试程序入门”介绍了调试环境及其基本操作，例如：如何从普通用户环境调用Debugger，如何退出Debugger。
- 第三章，“跟踪customer程序的逻辑”介绍了单个模块的customer程序的调试会话，（customer程序是提供给Debugger软件使用的）。为了熟悉customer程序的逻辑，你需要使用跟踪点，这是Debugger调试程序的一个特性。
- 第四章，“分析customer程序中的逻辑错误”，customer程序调试的继续。通过诊断程序的逻辑错误，你将学会用断点中断程序运行，以便得到程序的信息或与程序对话。
- 第五章，“多模块程序：cust_order”中有关于多模块程序的重要信息，例如：如何在普通用户环境下编译程序；如何在源码窗口显示指定的模块，如何在指定的模块中设置断点和跟踪点。
- 第六章，“跟踪cust_order程序的逻辑”，连续调试程序cust_order来增加断点和跟踪点的知识。本章介绍CALL命令，用来交互式地调试函数。
- 第七章，“分析运行时cust_order程序中出现的错误，”在调试cust_order程序过程中学会诊断运行时出现的错误。

参考部分

- 第八章，“调试环境”，详细介绍调试环境，并说明需要时，按调试的要求设置相应的环境。
- 第九章，“调试程序命令”，按字典顺序详细地列出了Debugger调试命令语法。

附录及错误信息部分

INFORMIX-4GL交互式调试程序指南附加如下内容：

- 附录A检查stores数据库，本指南中的样本程序是基于该数据库的。
- 附录B介绍可由INFORMIX—4GL和Debugger识别的环境变量。
- 附录C说明如何用Debugger调试程序，分析调用自定义C函数或 INFORMIX-ESQL/C 函数的4GL程序。
- 附录D讨论在本指南的调试对话中用过的样本程序。
- 附录E列出了INFORMIX—4GL和Debugger保留字。
- 附录F是ASCII表。

在附录后面，是错误信息部分。此部分列出了Debugger错误码，解释了它们的含意以及避免错误的方法。

（2）查看本指南

在样本调试对话的逻辑之后，需要知道INFORMIX—4GL语句的语法和特性。还要熟悉用户环境；特别是，如何修改及重新编译4GL程序模块（在指导部分，介绍如何编译多模块程序）。

（3）本指南中的约定

在这部分中讨论的约定将帮助你更好地使用本指南。

在整个指南中，Debugger的输出和命令将打印成计算机字体，如下所示：

```
view query_data
```

在指导部分中你应采取的动作按如下形式的标识（按键、输入命令等）：

▶（你要做什么）

用这种方式，使调试步骤区别于说明正文。

命令语法的约定除在此讨论外，在第九章中还要讨论。

CAPS UPPERCASE 这种字符串表示要求输入的关键字和任选项要如说明的那样输入。你可以用等价的缩写形式或小写形式代替。例如：

```
HELP TIMEDELAY
```

表示要输入命令：help timedelay

*italic*小写斜体字母的字符串表示该项必须用一个值代替，例如：

DATABASE *database_name*

表示在关键字database后面要输入一个指定的数据库名。语法后面的说明及注释描述了该项的类别。

〔 〕括号中是任选项。在命令行中不要用括号（除非说明数组元素）。如：

APPLICATION 〔DEVICE〕

意思是必须输入application，但也可以带任选项输入application device。每对括号表示不同的条件。如果一个关键字后面跟着几个由括号分开的任选项，则表示这些任选项可以全用，或用其中的几个，也可以都不用。例如：

LIST 〔BREAK〕 〔TRACE〕 〔DISPLAY〕

表示可以只单独输入list，或输入list break，或list后面跟这三个任选项的任意组合。

括号嵌套表示其中的任选项相关。只有当选择了外括号中的任选项，才能用内括

号中的任选项。例如：

BREAK [[module.]lineno]

意思是不能只用module.这个前缀，除非也给lineno一个值。

| 如果一条竖线将一对括号中的项分开，表示可以只选其中的一个任选项。例如：

CONTINUE [INTERRUPT | QUIT]

表示可以只单独输入continue，或continue后面跟interrupt或quit，但不能两者一起用。

{ } 一列任选项用花括号括起来，表示必须选用其中的一个任选项。在命令行中不要有花括号（既不要花括号，也不要隔离任选项表的竖线）。例如：

TRUE [ON | OFF] { AUTOTOGGLE | DISPLAYSTOPS |
EXITSOURCE | PRINTDELAY | SOURCETRACE }…

表示必须至少输入花括号中由AUTOTOGGLE开始，SOURCETRACE结尾表中的一个任选项。（在末尾的三个点，表示这个表后面还可以再附加些任选项，这将在后面解释）。

花括号还有一个特殊用法。必须把ALIAS,BREAK或TRACE命令中的命令字符串用花括号括起，它们的语法表示为：{ commands }或{ cmd_str[;cmd_str…] }。

_ 下划线是缺省任选项。在前面的例子中，下划线表示" ON"是缺省选项，即为当调用TURN命令时，如没有指定OFF或ON，则Debugger自动选择ON。（其中竖线分隔表示在同一个TURN命令中不能同时用ON和OFF。）

… 在括号中的三个点表示可以象前面的项那样重复任选项。例如：

CALL function ([arg [, arg…]])

它表示函数可以有一个任选参数，也可以再增加参数，参数间用逗号隔开。

在通常例子中，三个点单独出现在一行，表示在这行的前后命令之间可以有不定数量的其它命令。

样本程序和演示数据库

INFORMIX—4GL交互式调试程序软件包包含了在本指南的指导章节中你需要用于完成调试会话的所有文件。这些文件与Debugger程序文件在安装Debugger时要按照安装文件中的说明拷贝到一个目录下。

在第二章中介绍的customer程序是一个单模块程序，它只用一个屏幕格式。下面的文件需要建立为可执行文件。

customer.4gl
customer.per
custhelp.ex

cust_order程序将在第五章中介绍。它是个多模块程序，且使用两个屏幕格式。下面的文件需要建立为可执行文件：

customer.4gl
main.4gl
order.4gl

orderform.per

供调试程序用的其它文件有：

- 结言中用于INFORMIX—4GL交互式调试的程序为：

  orderinfo.4gl

- 附录C，“调用C函数”中用的4GL和c语言源程序，名为：

  r_globals.4gl
  r_main.4gl
  fgiusr.c
  getrand.c
  initrand.c

相应地，样本程序使用同样的数据库stores它包含有体育用品的批发商信息。stores数据库既用于INFORMIX—4GL快速开发系统，也用于Debugger软件包，它含有六个表：

customer
orders
items
stock
manufact
state

附录A将讨论数据库的结构和数据内容。

（1）建立stores数据库副本

在指导部分开始前，应将数据库做一个备份，如下所示：

①在你的工作目录下建一个目录供调试对话用。（当你调试完后，可以删除这个目录和它的内容。

②这个新目录必须包含在PATH变量中，系统将在其中的目录清单中查找可执行文件。（详看附录B）。

③将新建的目录变为当前目录，并运行dbdemo程序（由Debugger提供）建立一个演示数据库。

dbdemo

这个命令创建一个名为stores.dbs的子目录，并将数据库文件拷贝到该目录中。

（2）恢复数据库

当你对本指南中的样本程序熟悉后 便可以修改数据库的内容（删除一个顾客，增加一项订货等）。这结果使你看到与指导部分中的数据不同。在任何时候，都可以重新执行dbdemo程序来恢复原来的内容。

设置操作环境

要运行Debugger，操作环境必须完全设置成INFORMIX—4GL要求的。附录B中将介绍用于控制环境的环境变量。

如附录中所介绍的那样，Debugger能识别DBSRC环境变量，而INFORMIX—4GL却不能。在调试对话中，这个变量可以用作目录中寻找路径的参数。你不必为了用Debugger而特别设定DBSRC。

第一章 绪 论

1.1 本章概述

本章介绍关于INFORMIX—4GL交互式调试程序的重要概念，包括：

- 什么是调试程序
- 如何用Debugger学到由INFORMIX—4GL快速开发系统创建程序的更多知识。
- Debugger的主要命令及其作用。

INFORMIX—4GL交互式调试程序的使用将在第二章中介绍。

1.2 Debugger介绍

INFORMIX—4GL交互式调试程序是用来与INFORMIX—4GL程序运行时对话的工具。可用Debugger做以下工作：

- 使你很快熟悉别人写的程序或程序段。
- 分析程序中错误的原因。
- 学习INFORMIX—4GL语言的更多的功能。

例如，如果你怀疑程序的输出不对，而又没有Debugger，那么你只能做如下工作：

- 检查你的源程序
- 在纸上用数据跟踪执行程序，这通常叫桌上检查
- 在程序中设置打印语句DISPLAY，显示在程序运行中不同地方的变量值。

Debugger将使你可以很容易地检验INFORMIX—4GL程序工作是否正确，并使你检查和确定错误比用通常的方法更有效。Debugger通常用于发现逻辑错误（这种错误使程序产生不期望的结果）和致命错误（这种错误防碍程序继续运行）。

下面明确地指出Debugger的功能：

- 让程序在断点中断，然后继续运行程序
- 当程序运行时，按需要检查源程序代码，一次执行几行程序。
- 修改程序变量的值，并用新值重新运行程序。
- 在程序指定的行上，或函数变量，或变量值变化处，设置跟踪点来监视程序运行的情况。
- 在程序指定的行上或函数或变量值变化时，或当特定条件为真时，设置断点中断程序的运行。

INFORMIX—4GL交互式调试程序是源语言调试程序。除INFORMIX—4GL以外不必再了解别的语言就可以用Debugger。既可在INFORMIX—4GL用户环境，也可在命令行上调用Debugger。

1.3 使用Debugger

下面的节中介绍Debugger的主要特性及功能，并图示了用Debugger学习INFORMIX—4GL程序的方法。

1.3.1 Debugger屏幕

为了方便与程序对话，Debugger将终端环境分成如下两个屏幕：

- 应用屏幕
- 调试屏幕

应用屏幕

应用屏幕是用来显示INFORMIX—4GL程序的输入和输出的。程序运行完全如同在Debugger之外运行一样，可以进行菜单选择，输入和修改数据，并能象通常那样回应提示信息。

例如，下面的图说明customer程序在调试对话过程中的应用屏幕的外部特征。customer程序将在第二章中给出。选择Query项后，在屏幕表格中输入查询模式：

```
CUSTOMER:  Add  Query  Modify  Delete  Exit
Search for a customer.
------------------------------------------------------------

                      CUSTOMER FORM

   Number: [105 ]

 First Name: [            ] Last Name: [            ]

  Company: [            ]

  Address: [            ]
           [            ]

     City: [            ]

    State: [            ] Zipcode: [            ]

Telephone: [            ]
```

图1-1 Customer程序的应用屏幕

调试屏幕

为了监视源代码，为了与运行程序进行对话，就要中断程序运行，并显示调试屏幕。可在任何时候，按Interrupt键中断程序运行，并显示调试屏幕。

如果现在按了中断键，中断customer程序的运行，调试屏幕显示如下：

```
175         SLEEP 3
176
177         MESSAGE ""
178
179         CONSTRUCT where_clause on customer.* FROM customer_num,
180         fname, lname, company, address1, address2, city, state,
181         zipcode, phone
182
183           IF int flag THEN
(customer.4gl: query_data)
-------------------------------------------------------------------
$run
Stopped in query_data at line 179 in module " customer.4gl"
$
```

图1-2　Customer程序的调试屏幕

源码窗口

Debugger将调试屏幕分隔成两部分窗口进行工作，上部或源码窗口用来显示源代码，当中断运行时，源码窗口显示当前运行的程序段。一条亮线指示恢复运行后下一条要执行的语句。

可以用控制键或向上箭头和向下键头键将源码窗口上下滚动。例如，你可将屏幕滚动到显示MAIN段。

```
□10
 11   MAIN
 12
 13     DEFER INTERRUPT
 14
 15     OPEN FORM cust_form FROM "customer"
 16
 17     DISPLAY FORM cust_form
(customer.4gl: main)
$run
Stopped in query_data at line 179 in module "customer.4gl"
$view
```

图1-3　滚动源模块

命令窗口

下部或命令窗口用来输入Debugger命令。这些命令的输出也显示在此窗口中。当你第一次调用Debugger，或当你中断程序运行时，光标指在这个窗口的 $提示符下，Debugger等待输入。

例如，如果你输入FUNCTIONS命令，Debugger将在命令窗口列出在这个程序中所有的用户自定义函数。你可用与源码窗口同样的键来滚动命令窗口。

```
   10
   11  MAIN
   12
   13     DEFER INTERRUPT
   14
   15     OPEN FORM cust_form FROM "customer"
   16
   17     DISPLAY FORM cust_form
   18
(customer.4gl: main)

Stopped in query_data at line 179 in module "customer.4gl"
$ view
$ functions
change_data
delete_row
enter_row
main
query_data
show_menu
$
```

图 1-4　在命令窗口上的FUNCTION命令输出

Debugger提供大量参数，使你可以很容易地控制调试和应用屏幕之间的对话。例如，你可以告诉Debugger在程序运行时在源码窗口逐语句进行高亮度显示，也可以要求应用屏幕显示程序的输出，或只要求输入时显示。

在一个终端上，你可以同时既显示应用屏幕又显示调试屏幕。如果有两个终端，设定相同的TERMCAP，便可以给每个屏幕分配一个终端了。

1.3.2　跟踪点

设置跟踪点，可用来监视特定函数或语句的运行，或监视变量值的变化。跟踪点是用来跟踪陌生程序的有用工具。例如，如果你怀疑函数返回值不对时，它也是很有价值的调试工具。

下面的例子说明如何设置跟踪点监视cust_order程序中的get_stock 函数的运行。cust_order程序将在第五章中介绍。跟踪点用下面的Debugger命令设置：

```
trace get_stock
```

```
   21        CLEAR FORM
   22        ERROR "Order input aborted" ATTRIBUTE (RED, REVERSE)
   23        RETURN
   24     END IF
   25     INPUT ARRAY p_items FROM s_items.*
   26        BEFORE FIELD stock_num
   27        MESSAGE "Press ESC to write order "
```

```
28          DISPLAY "Enter a stock number or press CTRL-B to scan stock list"
( order.4gl; add_order )
$ trace get_stock
( 1 ) trace in function get_stock [ order.4gl ]
        scope function:  get_stock
$ run
Enter get_stock() from add_order line 45
Return ( 2, "HRO", "baseball    ", $126.00 ) from get_stock at line 195
```

图 1-5　cust_order程序跟踪点的输出

跟踪点输出调用get_stock 函数的程序行号及被调用函数的名字。当get_stock运行完后，跟踪点输出结束运行时的行号，也显示get_stock函数给调用程序返回的值。

1.3.3　断点

当预先设置的条件满足，断点就会使程序中断运行。这些条件包括：当一条指定的语句或函数运行，或当一个变量的值改变。你也能设置IF条件。

当中断运行时，你可以做这些工作：查看变量的当前值，或列出当前语句所调用的函数。你可以改变变量的值，并用新值接着运行程序。你还可以在要仔细检查的程序段设置断点，在程序运行到断点之后逐一语句地执行程序。程序运行到断点之后，必须发一条特定的Debugger命令，程序才能接着运行。

下一个例子是断点设在当cust_order程序中的全程记录成员p_customer.customer_num的值为107时中断程序运行。

这个断点用下面的命令设置

```
        break IF p_customer.customer_num=107
   83   DECLARE customer_set SCROLL CURSOR FOR statement_1
   84
   85   OPEN customer_set
   86   FETCH FIRST customer_set INTO p_customer.*
   87   IF status=NOTFOUND THEN
   88      LET exist=FALSE
   89   ELSE
   90      LET exist=TRUE
   91      DISPLAY BY NAME p_customer.* ATTRIBUTE ( MAGENTA )
( main.4gl; query customer )
$ break if p_customer.customer_num=107
( 1 ) break
          if:  p=customer.customer_num=107
$ run
Stopped in query_customer at line 87 in module "main.4gl"
$
```

图 1-6　cust_order程序中的断点输出

1.3.4 继续运行程序

debugger的一个不可缺少的功能是能够使程序从中断点继续运行。程序中断运行的原因是由于到达断点或是由于按了Interrupt键。

CONTINUE命令使程序从断点处继续运行。当你输入了CONTINUE命令后，只有在到达断点或输入了中断键，程序才会中断运行。例如，如果你在customer程序中的CONSTRUCT语句中断程序运行，如图1-1所示，现在可以用CONTINUE命令继续操作。应用屏幕重显查询步骤：

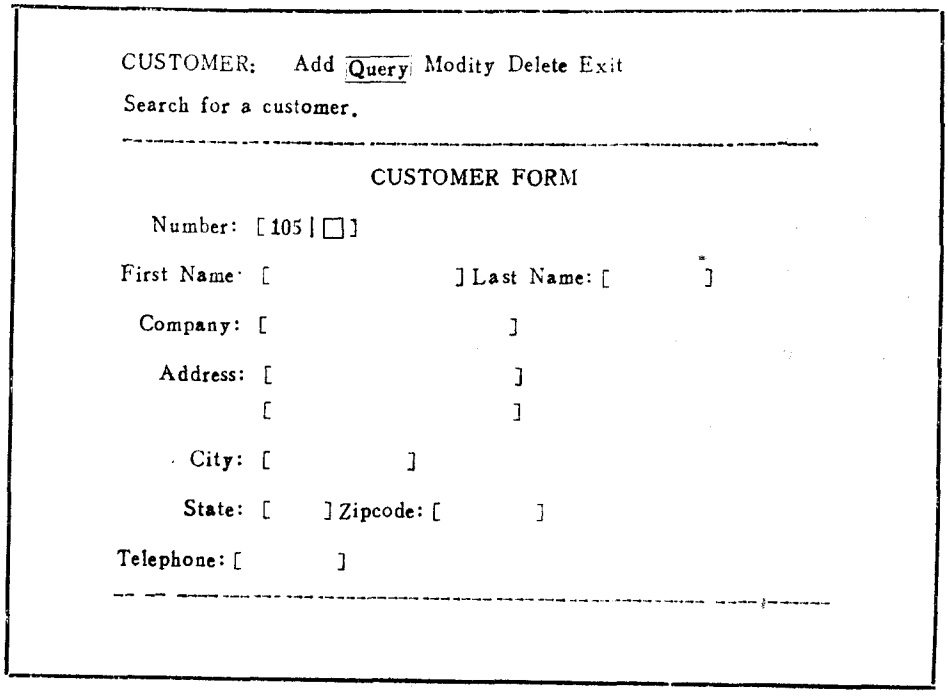

图1-7 用CONTINCE继续运行

与CONTINUE命令不同，可用STEP命令逐条语句地执行程序，或根据你的选择一次执行几行。逐条执行4GL语句是使你熟悉陌生程序的一个很好的方法。当用STEP命令时，Debugger的光标在源码窗口中指示下一条该执行的语句。命令窗口显示刚执行的语句的行号和函数名。

下面的例子是在cust_order程序发生中断的断点之后，用STEP命令执行下一条程序。

```
83  DECLARE customer_set SCROLL CURSOR FOR statement_1
84
85  OPEN customer_set
86  FETCH FIRST customer_set INTO p_customer.*
87  IF status=NOTFOUND THEN
88    LET exist=FALSE
```

```
   89  ELSE
   90    LET exist=TRUE
   91    DISPLAY BY NAME p_customer.*ATTRIBUTE(MAGENTA)
(main.4gl:query_customer)
$break if p_customer.customer_num=107
(1)break
          if:p_customer.customer_num=107
$run
Stopped in query_customer at line 87 in modole "main.4gl"
setp
Stoppped in query_customer at line 90 in module "main.4gl"
$□
```

图1-8　逐一语句运行

1.3.5　查看程序中的变量值

Debugger提供了两个简单方便的命令来查看程序变量的值。没有Debugger，就要在代码中分别设置DISPLAY语句来查看变量的值。

DUMP命令可以显示局部和全程变量的值及当前运行函数模块的变量值。如果指明了GLOBALS选项，DUMP命令将列出所有用户定义的全程变量的值及其模块变量的值，如INFORMIX— 4 GL全程变量status和SQLCA记录成员的值。

下图说明用DUMP命令显示当cust_order程序中运行renum_items函数时所有局部变量的当前值。

```
   147
   148  FUNCTION renum_item()
   149      DEFINE pa_curr, pa_total, sc_curr, sc_total, k INTEGER
   150
   152      LET pa_curr=arr_curr()
   151      LET pa_total=arr_count()
   153      LET sc_curr=scr_line()
   154      LET sc_total=4
   155      FOR k=pa_curr TO pa_total
(order.4gl:renum_items)

Stopped in renum_items at line 155 in module "order.4gl"
$dump
DUMPING LOCAL VARIABLES OF FUNCTION [renum_items]
   pa_curr=3
   pa_total=3
   sc_curr=3
   sc_total=4
   k=0
$□
```

图1-9　DUMP命令

PRINT命令允许你显示活动函数的单个变量的值。用这个单独的命令可以打印简单变量的值、程序记录的所有元素的值，或程序数组的所有元素的值。下面两个图是用PRINT命令显示的customer程序的标准输出。第一个例子，PRINT命令在命令窗口打印字符串sql_stmt的值：

```
202         LET exist=TRUE
203
204         DISPLAY BY NAME p_customer.*
205
206         PROMPT "Enter 'y' to select this customer",
207                 "or RETURN to view next customer:"
208                 FOR CHAR answer
209
210         IF answer="y"THEN
(customer.4gl:query_data)
$run
Stopped in query_data at line 206 in module "customer.4gl"
$print sql_stms
customer.4gl: query_data.sql_stmt="SELECT * FROM customer where customer.lname
matches"*son"
$□
```

图 1-10　打印变量的值

下一个例子是用PRINT命令将程序记录p_customer的元素的值保留在一个文件中。这个文件名为print1，赋给记录的值为Frank Albertson。下面的命令将产生这个文件：

```
print p_customer》print1
```

```
global:p_customer=record
    customer_num=114
    fname="Frank      "
    lname="Albertson  "
    company="Sporting place    "
    address1="947 waverly place "
    address2=(null)
    city="Redwood City  "
    stase="CA"
    zipcode="94062"
    phone="415-886-6677  "
end record
```

图 1-11　将程序记录的元素的值写到一个文件中