C20-1652-0

IBM Data Processing Techniques

IBM Operating System/360

TESTRAN User's Guide

This document presents guidelines to the effective use of program testing and debugging facilities provided by the test translator (TESTRAN) of Operating System/360.

Included are guidelines to writing the TESTRAN macros, assembling the problem program, executing the program in a test environment, and, finally, deleting TESTRAN from a debugged program. A sample program and its associated TESTRAN output serve as a vehicle for the discussion.

Programming

An understanding of the information in the following publications is prerequisite to use of this manual:

- IBM Operating System/360 Assembler Language (C28-6514)
- IBM Operating System/360 Linkage Editor (C28-6538)
- IBM Operating System/360 Job Control Language (C28-6539)

Reference is also made to IBM Operating System/360 Control Program Services (C28-6541).

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to IBM, Technical Publications Department, 112 East Post Road, White Plains, N. Y. 10601

CONTENTS

INTRODUCTION

The test translator (TESTRAN) is a program testing and debugging aid for OS/360 assembler language programs. The availability of TESTRAN in the operating system is determined at system generation time (SYSGEN).

TESTRAN consists of three parts:

1. TESTRAN macro instructions
2. TESTRAN interpreter
3. TESTRAN editor

The TESTRAN macros are the user's means of indicating (1) when testing is to begin, (2) where in the program testing is to take place, and (3) what tests are to be performed. The macros are expanded at assembly time from their macro definitions, which are included in the system macro library at SYSGEN. The expanded macro instructions are assembled into a TESTRAN interpreter action (TIA) table, in the form of a control section separate from the problem program control sections.

To execute the problem program with TESTRAN, the user must link-edit the problem program and TIA table CSECT's together, to form a single load module, which can then be fetched into main storage.

The TESTRAN interpreter operates in the supervisor state at program execution time. At appropriate times during execution of the problem program, the TESTRAN interpreter honors the test requests which have been coded in the TIA table. The interpreter routines are entered via SVC interruptions and LINK macro instructions, and are executed out of line from, but in succession with, the problem program. Output from the TESTRAN interpreter is placed on a system data set (SYSTEST) for later editing and printing by the TESTRAN editor. The functions of the interpreter are completed either when a TEST CLOSE statement is encountered or when the problem program reaches end-of-task.

The TESTRAN editor is a processor that operates in the problem program state. After completion of the user's program, the TESTRAN editor must be executed in order to print the test output created by the TESTRAN interpreter. This may be done immediately, or at a later time, since execution of the editor is a separate job or job step.

The editor reads the data from SYSTEST, and determines from a selection code whether a record is to be processed or skipped. If the record is to be processed, the editor provides proper headings, applies symbolic labels, converts the data to proper format, and writes the record onto the system print (SYSPRINT) data set.

CHAPTER 1: HOW TESTRAN OPERATES

The following section presents a brief explanation of TESTRAN from an internal viewpoint. Its purpose is to give the user some knowledge of how TESTRAN performs its functions, as this will help in understanding the procedure necessary to use TESTRAN effectively.

TESTRAN Macro Instructions and the TIA Table

The TESTRAN macro instructions can be grouped into five basic macros: TEST, TRACE, DUMP, SET, and GO. There are, in all, 23 variations of these five macros. (Refer to Operating System/360 Control Program Services, C28-6541, for a full description of the TESTRAN macros.) The TEST OPEN, TEST AT, and GO BACK macros have a special significance. The TEST OPEN macro initiates testing; the TEST AT macro indicates where testing is to be performed; and the GO BACK macro is used to return control to the problem program after a series of tests is completed.

TIA TABLE

The TIA table is constructed from the assembler's expansion of the TESTRAN macro instructions, according to their macro definitions which are contained in the system macro library. The table consists entirely of constants. Each macro instruction entry into the TIA table has a specific format. The entries in the table are in the same sequence as the source macro instructions that caused them to be created. Except for the TEST OPEN entry, the TIA table is nonexecutable.

TEST OPEN MACRO

The TEST OPEN macro instruction entry is always the first entry in the TIA table. Although this entry is identical to other macro instruction entries in that it contains various constants, the TEST OPEN entry has one important distinction. It is the only TIA table entry that is executable. The first byte of each TIA entry is the "entry type" that is specified in the macro definition; a different value is inserted for each of the 23 macro instruction types. Because the entry type byte for the TEST OPEN entry is the operation code for a supervisor call (SVC) instruction, this entry becomes executable.

TEST AT MACRO

The TEST AT macro instruction entry in the TIA table indicates where test services are to be performed in the problem program. At execution time, a TESTRAN interpreter routine inserts SVC instructions into the problem program at the places specified in the operands of the TEST AT macro instructions. The two bytes displaced by the SVC are saved in an interpreter table. When the problem program is executed, the inserted SVC instructions cause interruptions that pass control to the TESTRAN interpreter, which, in turn, initiates the performance of the test services.

GO BACK MACRO

The GO BACK macro instruction entry in the TIA table indicates that control is to be returned from the TESTRAN interpreter to the problem program. Since all testing takes place in the supervisor state, it is necessary to return control to the problem state before problem program execution can be continued. The GO BACK macro also specifies to what point in the problem program control is to be passed — to the next sequential instruction or to another point (specified in the GO BACK macro).

TESTRAN Interpreter

After assembly, the problem program and the TIA table must be "linkage-edited" and "fetched" into main storage. Testing of the problem program occurs in succession with the execution of the problem program, but "out of line" from it. That is, the test service routines of the interpreter are executed at the points within the problem program where the user has indicated he wanted them. The requested series of tests is performed, while the problem program execution is temporarily suspended, and at the conclusion of the series of test services, problem program execution is resumed.

At problem program execution time, control must be passed to the TEST OPEN entry in the TIA table in order to initiate testing. Since that entry is an SVC instruction, its execution causes an interruption which passes control to the TESTRAN interpreter. The interpreter inserts a TESTRAN SVC into the problem program at each point specified in the TEST AT macro instruction. The two bytes of the problem program displaced by the SVC instruction are stored in an interpreter table for later retrieval and execution.

Once the SVC insertions have been made, control passes to the entry point specified for the problem program, and execution of the problem program begins. When an SVC instruction is encountered in the problem program, the interruption processed for the SVC causes control to be passed to the interpreter's router routine, which determines the TEST AT macro instruction that caused the interruption and analyzes the TIA table entry which follows that TEST AT entry. The router then passes control to the proper service routine for the performance of the requested test service. When the current series of test requests in the TIA table is completed, the router routine passes control to the GO BACK routine. The GO BACK routine retrieves the two bytes of displaced problem program instruction from the interpreter table where they were stored when the TESTRAN SVC was inserted, reassembles the instruction at a remote location and executes it, and finally passes control back to the problem program. When the problem program reaches end-of-task, the function of the TESTRAN interpreter is completed.

The test output data, generated during the execution of the interpreter test service routines requested by the problem program under test, is written onto a system data set (SYSTEST) for storage and later editing by the TESTRAN editor.

TESTRAN Editor

The TESTRAN editor transcribes the information contained in the test data created by the interpreter into a printable output. It is a post-processor in that it functions only after the problem program whose test output it is to edit is terminated. The editor consists of discrete routines that read the test output data from the TESTRAN interpreter and select the records with the proper select (priority) codes for processing. The records are transcribed into the correct output format as determined by the type of interpreter routine that generated the data. Proper headings for the record type and available symbolic labels are written — with the data — onto the system print data set.

The editor analyzes the select code associated with a test output data record to determine whether or not it is to process that particular record. This determination is based upon the select code or codes specified in the job control EXEC statement for the TESTRAN editor job step. The priority or priorities indicated are compared with the select code in the record itself; those records whose select codes are acceptable are processed, and those whose select codes have not been specified are skipped. The actual select code for the job step is an integer (1 to 8), a blank, or the letter A. The select code in the record itself may be either an integer (1 to 8) or a blank. The TESTRAN editor is always a separate job or job step. It is not automatically executed, but must be called for in the same manner as any other processor.

CHAPTER 2: WRITING THE TESTRAN CONTROL SECTION

The purpose of this chapter is to present some guidelines to writing the TESTRAN macros. Figure 1 shows a subroutine to be tested. The subroutine, PRIMER, is designed to find prime numbers. Given an integer, X, PRIMER will find the next larger integer that is a prime number, and return this value to the calling program. It is desired to test the logic of the subroutine.

Inclusion of TESTRAN with the Problem Program

The first question is: "When should the TESTRAN macros be coded — before, during, or after the problem program has been coded?"

It is nearly impossible to write the TESTRAN macros before the subroutine has been written, since no labels have been defined in the subroutine. Writing the TESTRAN section during the coding of the subroutine has disadvantages in that the programmer must, in effect, write two programs at the same time. After the problem program has been coded, and all labels, data areas, and logic have been fully defined, it becomes a fairly simple matter to write a TESTRAN control section to go with it. Subroutines such as PRIMER present a special testing problem, since they depend on another program to call them. Often, the subroutine is completed before the calling program which will use it. The checkout and debugging of a subroutine, then, may depend on the completion of another program. This is not desirable, because (1) it means checking out two programs at the same time, which compounds the problems of debugging, and (2) the programmer who wrote the subroutine cannot turn his full attention to another project until the subroutine is checked out.

A classic solution to this problem has been to write a "dummy" calling program that passes constants to the subroutine and prints results of the calculations for a number of test cases. This has involved writing I/O statements for printing the results, as well as several CALL's, in order to present a significant number of test cases to the subroutine. This has allowed the subroutine to be debugged independently of the actual calling program.

This same basic procedure can be made easier through the use of TESTRAN. TESTRAN, because of its dynamic dump capability, can relieve the programmer of writing I/O statements to print results from the subroutine. Also, through use of the SET VARIABLE macro, a significant number of test cases can be generated to present to the subroutine. This is the procedure followed in Figure 1.

Explanation of Figure 1A

The TESTRAN macros for the subroutine were written upon completion of its coding. Since they were written before the "dummy" calling program, we will examine them first.

```
LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    E 01FEB66  3/30/66
                                    1          PRINT ON,NOGEN
                                    2 TTPRIME  TEST  OPEN
                                    3                *,***  IEGM04* - THIS MACRO ESTABLISHES CSECT TTPRIME
                                   11          TEST  AT,PRIMER
                                   12                *,***  IEGM09* - MACRO NUMBER 1 IN TTPRIME
                                   23          TRACE FLOW,PRIMER,ERR
                                   24                *,***  IEGM09* - MACRO NUMBER 2 IN TTPRIME
                                   38          DUMP  CHANGES,PRIMER,GOT+8
                                   39                *,***  IEGM09* - MACRO NUMBER 3 IN TTPRIME
                                   53          GO    BACK
                                   54                *,***  IEGM09* - MACRO NUMBER 4 IN TTPRIME
                                   63          TEST  AT,PRIMER+12
                                   64                *,***  IEGM09* - MACRO NUMBER 5 IN TTPRIME
                                   75          DUMP  PANEL,(G'10'),DATAM=F
                                   76                *,***  IEGM09* - MACRO NUMBER 6 IN TTPRIME
                                   89          GO    BACK
                                   90                *,***  IEGM09* - MACRO NUMBER 7 IN TTPRIME
                                   99          TEST  AT,(GOT+8,ERR)
                                  100                *,***  IEGM09* - MACRO NUMBER 8 IN TTPRIME
                                  112          TRACE STOP
                                  113                *,***  IEGM09* - MACRO NUMBER 9 IN TTPRIME
                                  120                *,***  IEGM41* - THIS TRACE STOP STOPS ALL TRACES
                                  124          DUMP PANEL,(G'9,11'),DATAM=F
                                  125                *,***  IEGM09* - MACRO NUMBER 10 IN TTPRIME
                                  140          GO    BACK
                                  141                *,***  IEGM09* - MACRO NUMBER 11 IN TTPRIME
000058                            150 PRIMER   CSECT
000058                            151          USING *,15
                                  152          SAVE  (14,12)
00005C 58C1 0000          00000   155          L     12,0(1)          R1 POINTS TO ADCON FOR X
000060 58AC 0000          00000   156          L     10,0(12)         PUT X IN R11
000064 12AA                       157          LTR   10,10
000066 47C0 F072          000CA   158          BC    12,ERR           IF X IS LE 0
00006A 1BBB                       159          SR    11,11            ZERO OUT R11
00006C 8CA0 0001          00001   160          SRDL  10,1             SHIFT LO-ORDER BIT INTO R11
000070 12BB                       161          LTR   11,11            Q--WAS BIT A ZERO
000072 4770 F02A          00082   162          BNZ   ODD              NO
000076 8CA0 001F          0001F   163          SRDL  10,31            YES--X IS EVEN--MOVE INTO R11
00007A 4AB0 F080          00088   164          AH    11,=H'1'         MAKE X ODD
00007E 47FC F032          00C8A   165          B     LOAD
000082 8CA0 001F          0001F   166 ODD      SRDL  10,31            MOVE X TO R11
000086 4AB0 F082          0008A   167          AH    11,=H'2'
00008A 4190 0003          00003   168 LOAD     LA    9,3              LOAD R9 WITH Y
00008E 1859                       169 AGAIN    LR    5,9              MOVE Y TO R5
000090 1C45                       170          MR    4,5              SQUARE Y
000092 195B                       171          CR    5,11             Q--IS Y**2 GT X
000094 4720 F060          00088   172          BH    GOT              YES--X IS PRIME
000098 4780 F058          00080   173          BE    INCR             X = Y**2--X NOT PRIME
00009C 183B                       174          LR    3,11                PREPARE
00009E 1B22                       175          SR    2,2                    TO DIVIDE
0000A0 1D29                       176          DR    2,9                  X/Y
0000A2 1222                       177          LTR   2,2                 Q)--REMAINDER
0000A4 4780 F058          00080   178          BZ    INCR             NO REMAINDER. X NOT PRIME
0000A8 4A90 F082          0008A   179          AH    9,=H'2'          REMAINDER--ADD 2 TO Y
0000AC 47F0 F036          0008E   180          B     AGAIN            TRY DIVISION BY NEW Y
0000B0 4ABB F082          0008A   181 INCR     AH    11,=H'2'         X WAS NOT PRIME--ADD 2 TO X
0000B4 47F0 F032          00C8A   182          B     LOAD             GO SEE IF X+2 IS PRIME
0000B8 58C1 0004          00004   183 GOT      L     12,4(1)          R1+4 POINTS TO RESULT FIELD ADCON
0000BC 50BC 0000          00000   184          ST    11,0(12)         STORE RESULT
                                  185          RETURN (14,12),RC=4    NORMAL RETURN FROM PRIMER
                                  189 ERR      RETURN (14,12),RC=0    ERROR EXIT FROM PRIMER
0000D8                            194          LTORG
0000D8 0001                       195                =H'1'
0000DA 0002                       196                =H'2'
000058                            197          END   PRIMER
```

Figure 1A. PRIMER

Name	Operation	Operand
TTPRIME	TEST	OPEN

A TEST OPEN macro must be the first TESTRAN macro encountered at assembly time. There must be a name in the name field, since this becomes the name of the TESTRAN CSECT which is generated. No optional operands were coded in this case, since all the additional parameters for TEST OPEN will come from the TESTRAN control section to be written in conjunction with the "dummy" calling program. Encountering this TEST OPEN macro at execution time causes SVC's to be placed at the locations specified in the following TEST AT macros (PRIMER,PRIMER+12,GOT+8,ERR).

Name	Operation	Operand
	TEST	AT,PRIMER

The TEST AT macro defines a procedure and requests TESTRAN services at a specific point or points in the problem program. In this case, when the subroutine is entered at location PRIMER, the TESTRAN interpreter honors the requests that follow the TEST AT,PRIMER macro.

Name	Operation	Operand
	TRACE	FLOW,PRIMER,ERR

This macro requests a flow trace of the entire subroutine (that is, from location PRIMER to location ERR). This trace produces output describing all branches that occur within the subroutine. This can be an invaluable debugging aid.

When the request is honored at execution time, the TESTRAN interpreter goes into "trace mode", that is, interpretive mode. Tracing activity requires the examination of each problem program instruction executed while the trace is active. This process is time-consuming, but the time required can be minimized by limiting the duration of the trace and the output the trace produces. The programmer can limit the duration of the trace by means of TRACE STOP macro instructions; when encountered at execution time, these macro instructions suspend tracing activities specified as being of no current interest to the programmer. The programmer can limit the output of the trace, within a general area of main storage, to only those areas of special interest by specifying each such area in a separate macro instruction. No more than ten traces (corresponding to ten TRACE macro instructions) can be active simultaneously.

The programmer must consider the following facts when interpreting the output of trace routines:

- Tracing is suspended when supervisor-state control program routines receive control, and is resumed when the control program relinquishes control.

- A trace that is active in an overlay segment is suspended if the segment is overlaid, and is not automatically resumed when the segment is reentered.

- A subroutine or program segment must contain its own TRACE macro instructions if it receives control from the control program through an asynchronous exit or through an ATTACH, LINK, or XCTL macro instruction. All traces active for a task are suspended upon execution of an ATTACH, LINK, or XCTL macro and upon execution of a RETURN macro instruction that terminates a task or program receiving control through one of these macro instructions.

- Tracing is performed only in storage areas associated with the problem program or with problem-state control program routines.

The range of addresses of those storage areas traced is determined by the "loaded addresses" assigned to the operands of the TRACE FLOW macro. Thus, TRACE FLOW,PRIMER,ERR will trace from the location in which the instruction named PRIMER is actually loaded through the location in which the instruction named ERR is actually loaded. The storage addresses at which the program is actually loaded are called "loaded addresses". In a problem program that is scatter-loaded, the range of addresses traced may vary unpredictably if the starting and ending addresses are in separate control sections. The following conditions may occur:

- Control sections that were not part of the range in the source program may be included in the range of loaded addresses.

- Control sections that were a part of the original range may be omitted from the range of loaded addresses.

- The control section containing the starting address may be loaded at a higher-numbered storage location than the control section containing the ending address. If this situation occurs, the macro instruction is ignored and a diagnostic message is inserted in the test output each time the macro instruction is encountered.

A program that is scatter-loaded should therefore include a separate macro instruction for each control section in which traces are to be recorded.

Name	Operation	Operand
	DUMP	CHANGES, PRIMER, GOT+8

The DUMP CHANGES macro is used here to record any modification of the program which might occur through an error. Anything that is modified at execution time, from the beginning of the subroutine (PRIMER) to the end (GOT+8), will be recorded. Although this will not prevent the subroutine from erroneously modifying itself (for example, by storing data over an instruction), it will at least give the programmer an indication as to what was changed. This information, used in conjunction with the trace output, would help him pinpoint the error. Or, a TRACE REFER macro could be included later to exactly pinpoint the instruction(s) that caused the modification.

Name	Operation	Operand
	GO	BACK

The GO BACK macro returns control to the problem program. In this case, since TRACEing has been requested, the TESTRAN interpreter

enters the TRACE mode. Thus, as the problem program is executed, the TRACE routine inspects each op-code to see whether it is a branch, and, if so, records the branch instruction, the addresses from and to which the branch is made, as well as the condition code at the time of the branch.

Name	Operation	Operand
	DUMP	PANEL,(G'10'),DATAM=F
	GO	BACK

At PRIMER+12, TESTRAN will dump general register 10, which should at this point contain the test value for X. DATAM=F specifies that register 10 contains fixed-point fullword data. The TESTRAN editor will convert the data to decimal on output. If DATAM is not specified, the register contents are formatted as four-byte hexadecimal.

GO BACK causes a return to the problem program. Note that TRACEing will resume, since no TRACE STOP has been requested.

Name	Operation	Operand
	TEST	AT,(GOT+8,ERR)

This macro instruction indicates the beginning of a test procedure to be executed at two locations in the problem program: GOT+8 and ERR. When the problem program reaches GOT+8 or ERR, the following three requests will be honored:

Name	Operation	Operand
	TRACE	STOP
	DUMP	PANEL,(G'9,11'),DATAM=F
	GO	BACK

The TRACE STOP macro causes the TESTRAN interpreter to leave the trace mode. That is, tracing is suspended until requested again.

The DUMP PANEL statement calls for a dump of general registers 9, 10, 11. The DATAM=F specification performs the same function as in the previous DUMP PANEL macro.

The GO BACK macro returns control to the problem program after executing the instruction that was replaced by TESTRAN's SVC. This is the end of the TESTRAN control section TTPRIME, since no more TESTRAN macros appear before the END card for the assembly.

Explanation of Figure 1B

The dummy calling program, CALLTEST, is designed specifically to supply the subroutine, PRIMER, with test data. The program relies on TESTRAN macros to generate this data. CALLTEST supplies two data areas: TESTVAR, which will be the test data supplied to the subroutine; and ANS, where the subroutine will place the result of its calculation. Also supplied is a save area, as dictated by linkage conventions. The program calls PRIMER, and TESTRAN then dumps the result from the subroutine.

Name	Operation	Operand
DATAGEN	TEST	OPEN,CALLTEST,TESTING,LOAD, OPTEST=(TTPRIME),SELECT=1

DATAGEN becomes the name of the TESTRAN control section.

```
LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  E 01FEB66  4/01/66
                                    1          PRINT ON,NOGEN
                                    2 ***************************************************
                                    3 *****THIS TESTRAN CSECT IS DESIGNED TO SUPPLY TEST DATA AND
                                    4 ********TO  DUMP RESULTS OF THE SUBROUTINE
                                    5 ***************************************************
                                    6          EXTRN TTPRIME
                                    7 DATAGEN  TEST  OPEN,CALLTEST,TESTING,LOAD,OPTEST=(TTPRIME),SELECT=1
                                    8                *,***  IEGM04* - THIS MACRO ESTABLISHES CSECT DATAGEN
                                   20          TEST  DEFINE,COUNTER,K1
                                   21                *,***  IEGM09* - MACRO NUMBER 1 IN DATAGEN
                                   31          TEST  AT,CALLTEST
                                   32                *,***  IEGM09* - MACRO NUMBER 2 IN DATAGEN
                                   43          SET   COUNTER,K1,=F'-3'
                                   44                *,***  IEGM09* - MACRO NUMBER 3 IN DATAGEN
                                   57          GO    BACK
                                   58                *,***  IEGM09* - MACRO NUMBER 4 IN DATAGEN
                                   67          TEST  AT,TTSVC1
                                   68                *,***  IEGM09* - MACRO NUMBER 5 IN DATAGEN
                                   79          TEST  ON,1,100,100,QUIT,COUNTER=K1
                                   80                *,***  IEGM09* - MACRO NUMBER 6 IN DATAGEN
                                   98          SET   VARIABLE,TESTVAR,K1
                                   99                *,***  IEGM09* - MACRO NUMBER 7 IN DATAGEN
                                  113          GO BACK
                                  114                *,***  IEGM09* - MACRO NUMBER 8 IN DATAGEN
                                  123          TEST  AT,TTSVC2
                                  124                *,***  IEGM09* - MACRO NUMBER 9 IN DATAGEN
                                  135          DUMP  DATA,TESTVAR,ANS+4
                                  136                *,***  IEGM09* - MACRO NUMBER 10 IN DATAGEN
                                  150          TEST  WHEN,K1,LT,=F'1',MESSAGE,DATAM=F
                                  151                *,***  IEGM09* - MACRO NUMBER 11 IN DATAGEN
                                  169          GO    BACK
                                  170                *,***  IEGM09* - MACRO NUMBER 12 IN DATAGEN
                                  179 MESSAGE  DUMP  PANEL,(G'15')
                                  180                *,***  IEGM09* - MACRO NUMBER 13 IN DATAGEN
                                  192          DUMP  COMMENT,'REGISTER 15 SHOULD CONTAIN 0'
                                  193                *,***  IEGM09* - MACRO NUMBER 14 IN DATAGEN
                                  209          GO    BACK
                                  210                *,***  IEGM09* - MACRO NUMBER 15 IN DATAGEN
                                  219 QUIT     DUMP  PANEL
                                  220                *,***  IEGM09* - MACRO NUMBER 16 IN DATAGEN
                                  231          GO    BACK,RET
                                  232                *,***  IEGM09* - MACRO NUMBER 17 IN DATAGEN
000000                            243 CALLTEST CSECT
                                  244 ***************
                                  245 *************** CALLTEST IS A TEST ROUTINE FOR PRIMER
                                  246 ***************
                                  247          SAVE  (14,12)
000000C4 05C0                     250          BALR  12,0
000000C6                          251          USING *,12
000000C6 50D0 C036        000FC   252          ST    13,MINE+4
000000CA 41D0 C032        000F8   253          LA    13,MINE
000000CE 07C0                     254 TTSVC1   NOPR  0
                                  255          CALL  PRIMER,(TESTVAR,ANS)
000000EA 47FC C008        000CE   268 TTSVC2   B     TTSVC1
000000F0                          269 TESTVAR  DS    F
~~000000F4~~                      ~~270 ANS~~      ~~DS~~    ~~F~~
000000F8                          271 MINE     DS    18F
00000140 58DC C036        000FC   272 RET      L     13,MINE+4
                                  273          RETURN (14,12)
                                  276          END
```

Figure 1B. CALLTEST

CALLTEST is the entry point to the problem program. At the conclusion of the TEST OPEN routine (that is, after SVC's have been inserted at the required places in the problem program), TESTRAN transfers control to location CALLTEST in the problem program.

TESTING is a page heading to be printed on each page of TESTRAN output.

LOAD specifies that the TESTRAN service routines (for example, DUMP, TRACE) are to be resident throughout problem program execution, rather than transient (loaded when needed). This requires more core storage than the transient mode (specified by LINK), but results in faster execution. TESTRAN will automatically default to transient (LINK) mode if sufficient core storage is not available for LOAD mode operation.

OPTEST=(TTPRIME) indicates that the TESTRAN control section, TTPRIME, is to be opened at the same time as DATAGEN. Notice that TTPRIME has been declared to be an external symbol (EXTRN TTPRIME). This is necessary because the subroutine and its TESTRAN macros are assembled separately from CALLTEST and its TESTRAN macros. This would be necessary for any external symbol that appeared in the TESTRAN macros. Similarly, TTPRIME or any other external reference must be a CSECT name or must appear as the operand of an ENTRY statement in another assembly.

SELECT=1 specifies that a selection code of 1 is to be attached to the output from this TESTRAN control section and from any other TESTRAN CSECT's opened at the same time (for example, TTPRIME). This selection code will be in force for all macros unless overridden in a particular macro. For example, a DUMP macro could specify SELECT=5, in which case the output from that macro would have a selection code of 5, rather than 1. Through the selection code, the user, at TESTRAN Editor time, can selectively print the output produced by the interpreter. This is done by supplying a parameter to the editor, specifying that records with certain selection codes are to be printed and that all others are to be skipped.

Name	Operation	Operand
	TEST	DEFINE,COUNTER,K1

This macro sets up a COUNTER, to be named K1, for use within the TESTRAN control section. K1 becomes the name of a data area defined by TESTRAN, and is initialized to zero (0).

Name	Operation	Operand
	TEST	AT,CALLTEST
	SET	COUNTER,K1,=F'-3'
	GO	BACK

The test service requested at location CALLTEST is the initialization of K1 to minus three. The third operand in a SET macro must be an address. In this example, a literal is specified, which results in an address. If the third operand is written as a self-defining value, TESTRAN interprets it as an address. For example, SET COUNTER, K1,1000 will not set the counter to the value 1000, but rather will transfer the contents of location 1000 to the counter.

Just before calling the subroutine, the program requests test services in order to generate test data.

Name	Operation	Operand
	TEST	AT,TTSVC1
	TEST	ON,1,100,100,QUIT,COUNTER=K1
	SET	VARIABLE,TESTVAR,K1
	GO	BACK

TEST ON,1,100,100,QUIT,COUNTER=K1 increments K1 by 1, tests to see whether K1 is between the limits specified by operands 2 and 3 (1 and 100), and, if it is, tests to see whether K1 is a multiple of operand 4 (100). If both conditions are true, TESTRAN transfers control to the macro named QUIT. Otherwise, control passes to the next sequential macro.

The SET VARIABLE macro is used to change the value of a problem program data area. In this case, TESTVAR is set to the current value of K1. This value changes before each CALL by virtue of the TEST ON macro.

The GO BACK macro returns control to the location following TTSVC1, where the problem program issues a CALL to PRIMER, and passes TESTVAR as a parameter.

Name	Operation	Operand
	TEST	AT,TTSVC2
	DUMP	DATA,TESTVAR,ANS+4
	TEST	WHEN,K1,LT,=F'1',MESSAGE,DATAM=F
	GO	BACK

At TTSVC2 (that is, after the subroutine returns control to the calling program), TESTRAN will DUMP all data between TESTVAR and ANS+4. That is, the two data areas, TESTVAR and ANS, will be output and formatted according to their definitions (fullword).

The TEST WHEN macro transfers control to the TESTRAN macro named MESSAGE when counter K1 is less than (LT) 1 (=F'1'). If K1 is greater than or equal to 1, control passes to the next sequential macro — GO

BACK. The DATAM=F specifies that K1 is a fullword data area. This is necessary because K1's attributes do not appear in the symbol table, and, if not specified, TESTRAN assumes one-byte hexadecimal. (Note: This is necessary only on TEST WHEN. The DATAM keyword is illegal if operand 1 or 2 is a logical flag.) The GO BACK macro executes the instruction at TTSVC2 (B TTSVC1) and returns control to the program at location TTSVC1. Since TTSVC1 is a TEST AT location, control returns immediately to TESTRAN and the requested services are performed.

Name	Operation	Operand
MESSAGE	DUMP	PANEL,(G'15')
	DUMP	COMMENT,'REGISTER 15 SHOULD CONTAIN 0'.
	GO	BACK

The subroutine, PRIMER, will refuse to process data which is zero or negative. To indicate an error, it provides a return code of 0 in register 15. (The normal return from PRIMER places a return code of 4 in register 15.) The TEST WHEN macro, above, determines when the data supplied to PRIMER is zero or negative, and, when it is, passes control to MESSAGE.

At location MESSAGE, general register 15 is displayed, along with a comment indicating that register 15 should contain zero. If register 15 does not contain zero at this point, the logic of the subroutine did not recognize the erroneous data.

Name	Operation	Operand
QUIT	DUMP	PANEL
	GO	BACK,RET

When K1 contains 100, TESTRAN passes control to the macro named QUIT (see TEST ON above). At this point all registers are DUMPed.

The GO BACK macro returns control to location RET in the problem program, which terminates the job via a RETURN macro.

There is no TEST CLOSE macro in this example, because TEST CLOSE returns control to the TEST AT location, which in this case is an unconditional branch. The function of TEST CLOSE is to remove the TESTRAN SVC's from the problem program TEST AT locations. The assumption is that the programmer wishes to resume execution of his program without TESTRAN. Since CALLTEST is not designed to operate free from TESTRAN, a TEST CLOSE macro would perform no useful function, and is therefore unnecessary.

CHAPTER 3: GENERAL PROCEDURES FOR USING TESTRAN

After the problem program(s) and TESTRAN control section(s) have been written, they must be assembled. A program and its TESTRAN control section may be assembled either together or separately.

Assembling TESTRAN Macros with the Problem Program

If assembled with the problem program, the TESTRAN macros may be written out of line (as in Figure 1) or in line (that is, interspersed with the problem program instructions). In either case, the TIA tables produced will reside in a separate control section from the problem program. That is, whether the user chooses to write the source statements for TESTRAN in line or out of line, the final result is an out-of-line TESTRAN control section. Since this is true, and since TESTRAN requests will not be executed in line, there is no advantage to writing the macros in line with the problem program. The disadvantage, of course, is that, should the user wish later to reassemble his program without the TESTRAN macros, he must search through the source deck in order to remove the TESTRAN statements.

Assembling TESTRAN Macros Separately from the Problem Program

If the problem program is assembled separately from the TESTRAN macros, the user must follow the rules for external references and entry points. That is, any problem program symbol (label) which appears as the operand of a TESTRAN macro must appear in an EXTRN statement when the TESTRAN macros are assembled. These same symbols must be control section names or operands of an ENTRY statement in another assembly. In most cases, this is inconvenient. There are times, however, when it may be desirable to assemble the TESTRAN macros separately from the problem program. A user may wish to include TESTRAN with an object deck, for example, without reassembling the problem program. In this case, the operands of the TESTRAN macros must all be declared in EXTRN statements for the TESTRAN control section, and must appear as ENTRY symbols in the external symbol dictionary (ESD) of the problem program. Since the user has no way to add symbols to the ESD without reassembly, he is forced to use address adjustment to form the operands for the TESTRAN macros. This is, at best, time-consuming.

Writing the TESTRAN macros out of line and assembling them with the problem program appears to be the simplest and most convenient method for using TESTRAN.

TESTRAN Job Flow and Job Control Language

Figure 2 is a general TESTRAN job flow in flowchart form. Figure 3 is an example of the job control language to assemble, linkage-edit, and execute the two programs of the previous example, and to edit the output produced by TESTRAN.

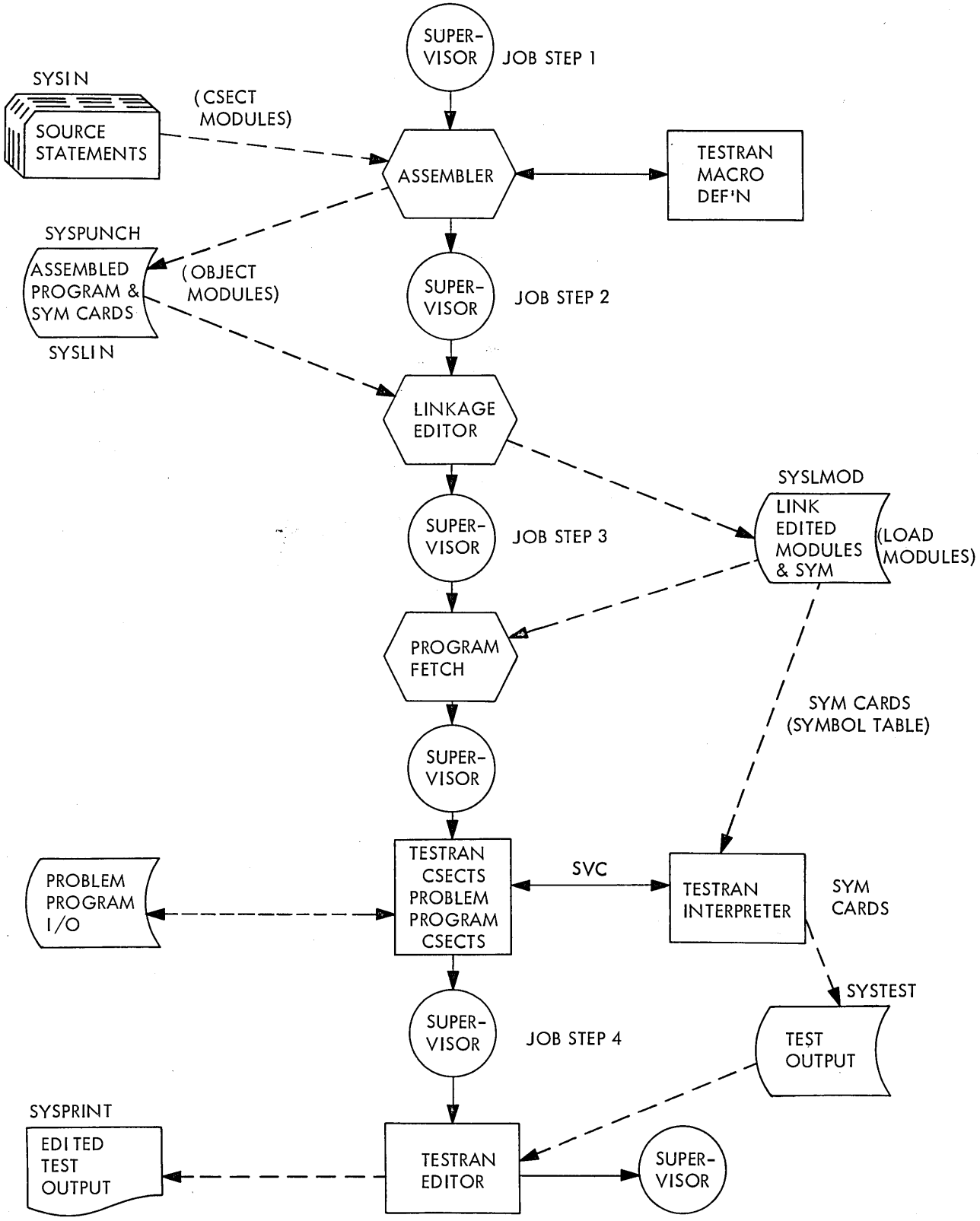

Figure 2. General TESTRAN job flow

The computing system used for this example requires at least three tape drives and one 2311 disk drive. The unit designations (2400, 2311) used on the DD cards in the following examples are the generic device names defined in the Job Control Language manual.

```
//JOBB   JOB   1234,YOUNG,MSGLEVEL=1
//STEP1  EXEC  PGM=IETASM,PARM=TEST
//SYSUT1 DD    UNIT=2400,LABEL=(,NL)
//SYSUT2 DD    UNIT=2400,LABEL=(,NL)
//SYSUT3 DD    UNIT=2400,LABEL=(,NL)
//SYSLIB DD    DSNAME=SYS1.MACLIB,UNIT=2311,DISP=OLD,VOLUME=SER=111111
//SYSPUNCH DD DSNAME=OBJ1,UNIT=2311,DISP=(NEW,PASS),                    X
//             SPACE=(TRK,(10,10)),VOLUME=SER=111111
//SYSPRINT DD  SYSOUT=A
//SYSIN  DD    *

---SOURCE DECK FOR PRIMER


//STEP1A EXEC  PGM=IETASM,PARM=TEST
//SYSUT1 DD    UNIT=2400,LABEL=(,NL)
//SYSUT2 DD    UNIT=2400,LABEL=(,NL)
//SYSUT3 DD    UNIT=2400,LABEL=(,NL)
//SYSLIB DD    DSNAME=SYS1.MACLIB,UNIT=2311,DISP=OLD,VOLUME=SER=111111
//SYSPUNCH DD DSNAME=OBJ2,UNIT=2311,DISP=(NEW,PASS),                    X
//             SPACE=(TRK,(10,10)),VOLUME=SER=111111
//SYSPRINT DD SYSOUT=A
//SYSIN  DD    *

-----  SOURCE DECK FOR CALLTEST


//STEP2  EXEC  PGM=LINKEDIT,PARM='XREF,TEST'
//SYSUT1 DD    DSNAME=UTIL,UNIT=2311,SPACE=(TRK,(40,10))
//SYSLMOD DD   DSNAME=GOFIL(ABC),UNIT=2311,SPACE=(TRK,(40,10,1)),       X
//             DISP=(NEW,PASS)
//SYSPRINT DD  SYSOUT=A
//DD1    DD  DSNAME=OBJ1,UNIT=2311,DISP=(OLD,DELETE),VOLUME=SER=111111,X
//             DCB=(RECFM=F,BLKSIZE=80)
//DD2    DD  DSNAME=OBJ2,UNIT=2311,DISP=(OLD,DELETE),VOLUME=SER=111111,X
//             DCB=(RECFM=F,BLKSIZE=80)
//SYSLIN DD    *
         INCLUDE DD1
         INCLUDE DD2
         ENTRY DATAGEN

//STEP3  EXEC  PGM=*.STEP2.SYSLMOD
//SYSTEST DD   DSNAME=TESTOUT,UNIT=2400,DISP=(NEW,PASS),LABEL=(,NL),    X
//             VOLUME=SER=A11111
//STEP4  EXEC  PGM=IEGTTEST,PARM=TA
//SYSTEST DD   DSNAME=TESTOUT,UNIT=2400,DISP=(OLD,DELETE),LABEL=(,NL), X
//             VOLUME=SER=A11111
//SYSUT1 DD    DSNAME=WORK,UNIT=2311,SPACE=(TRK,(40,10))
//SYSPRINT DD  SYSOUT=A
```

Figure 3. Job Control Language for using TESTRAN

ASSEMBLY (STEPS 1 AND 1A)

The assembly procedure specifies TEST as a parameter to the assembler. This is a request to the assembler to include the symbol table in the object modules produced for PRIMER and CALLTEST. This is necessary when the problem program is assembled if the user wishes the TESTRAN editor to perform automatic formatting of TESTRAN output.

The assembler requires three work data sets, SYSUT1, SYSUT2, and SYSUT3, which, in the example, are specified as tape. The object modules produced by the assembler are given the names OBJ1 and OBJ2, and are PASSed to the Linkage Editor job step.

LINKAGE EDITOR PROCEDURE (STEP2)

The two object modules, containing CALLTEST, PRIMER, and their respective TESTRAN control sections, are link-edited together.

The TEST parameter in the EXEC card for STEP2 specifies that Linkage Editor is to process the symbol table (SYM) cards that were produced by the assembler. If TEST is not specified, the SYM cards are ignored.

The user specifies that the entry point to the load module will be DATAGEN. When the program is loaded, the first instruction executed is the SVC for TEST OPEN, which will initiate testing.

PROBLEM PROGRAM EXECUTION (STEP3)

The user must provide an output data set for TESTRAN when his program is executed. This is done by including a DD card for SYSTEST, specifying a tape or direct access device.

TESTRAN EDITOR PROCEDURE (STEP4)

Upon termination of program execution, the user may execute the TESTRAN Editor (IEGTTEDT). Three DD cards are necessary: SYSTEST, SYSPRINT, and SYSUT1. SYSTEST is, of course, the data set produced in STEP3. SYSPRINT is the systems print data set. SYSUT1 is a direct access device to be used by the editor as an intermediate work file. The parameter TA on the EXEC card specifies that all data is to be edited, regardless of its selection code.

At the conclusion of STEP4, SYSUT1 will be deleted from the VTOC of the 2311. OBJ1 and OBJ2 were deleted following STEP2. GOFIL, the Partitioned Data Set created in STEP2, is automatically deleted at the end of STEP4, by virtue of the fact that its disposition in STEP2 was NEW, PASS, and no further disposition was found.

CHAPTER 4: TESTRAN OUTPUT

On the following pages are presented the first six pages of the output produced by TESTRAN for the programs of Figure 1. In the accompanying explanation of selected lines, the circled numbers correspond to the numbers on the listing. Refer to IBM Operating System/360 Control Program Services (C28-6541) for a complete and detailed description of TESTRAN output.

(1) The output from TEST OPEN. The maximum number of pages and maximum number of statements are the default figures established at system generation for this particular system.

(2) Indicates the execution of an inserted TESTRAN SVC. The TESTRAN control section referenced is TTPRIME and the SVC is at relative location 000058 (loaded address is 5778) in control section PRIMER. Note that this relative location is the location noted on the assembly listing.

(3) Indicates that a TRACE FLOW has been initiated in TTPRIME. The trace is to be active between relative locations 58_{16} and CA_{16} in control section PRIMER.

(4) DUMP CHANGES output. Since this is the first time that this macro has been encountered, the entire area of core is dumped. Each output group consists of two or three lines. The first line contains the relative location of the dumped area (for example, 58), and any labels that appeared in the assembly (for example, ODD,AGAIN). The second line contains the loaded address of the area (5778) and the data, formatted according to type. In this case the data consisted of instructions, and so was formatted accordingly. The third line indicates SVC 26 at those locations where a TESTRAN SVC was inserted.

(5) Indicates that a trace event has occurred at relative location 64 in PRIMER. The event was the execution of an SVC 26, as indicated on the second line. Also printed are the contents of the standard linkage registers, 0, 1, 14, and 15. SVC 26 is the SVC inserted at TEST OPEN time as a result of the macro TEST AT,PRIMER+12. Note that relative location 64 is equivalent to PRIMER+12.

(6) Output of the DUMP PANEL macro in TTPRIME which requested that general register 10 be dumped.

(7) Indicates the occurrence of a trace event in PRIMER. A branch (BC C0 F 072) has occurred from relative location 000066 to relative location 0000CA. The condition code was 1 (CC=1). Also displayed is general register 15, since that is the base register for the branch address.

```
(1)  TESTING                    TESTRAN OUTPUT                DATE 66/084              TIME 00/00          PAGE    1
1) MACRO ID 000, TEST OPEN   , TESTRAN CONTROL SECTION = DATAGEN , IDENTIFICATION   TESTING

 MAXIMUM NUMBER OF PAGES 150, MAXIMUM NUMBER OF STATEMENTS 2500

(2)
AT LOCATION (PRIMER  ) 000058 005778  ENTER  TTPRIME

(3)
1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000058 005778 TO (PRIMER  ) 0000CA 0057EA, STARTED

(4)
1) MACRO ID 003, DUMP CHANGES
  0058
005778     STM  EC D 00C       L    C1 0 000       L    AC 0 000       LTR  AA              BC   C0 F 072
           SVC  26                                                    SVC  26
  006A
00578A     SR   BB            SRDL A0 0 001       LTR  BB             BC   70 F 02A       SRDL A0 0 01F
  007A                                             ODD                                    LOAD
00579A     AH   B0 F 080       BC   F0 F 032       SRDL A0 0 01F       AH   B0 F 082       LA   90 0 003
  008E     AGAIN
0057AE     LR   59            MR   45            CR   58            BC   20 F 060       BC   80 F 058
  009C
0057BC     LR   3B            SR   22            DR   29            LTR  22            BC   80 F 058
  00A8                                             INCR                                   GOT
0057C8     AH   9C F 082       BC   F0 F 036       AH   B0 F 082       BC   F0 F 032       L    C1 0 004
  00BC
0057DC     ST   BC 0 000

(5)
1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000064 005784, CC=0
SVC 26              G'00' 0000003C G'01' 8000581C G'14' 400C582A G'15' 00005778

AT LOCATION (PRIMER  ) 000064 005784  ENTER  TTPRIME

(6)
1) MACRO ID 006, DUMP PANEL
G'10' -2
PSW  FF 0 5 0026 4 0 005786  CC=0  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

(7)
1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000066 005786 TO (PRIMER  ) 0000CA 0057EA, CC=1
BC   C0 F 072        G'15' 00005778

AT LOCATION (PRIMER  ) 0000CA 0057EA  ENTER  TTPRIME

(8)
1) MACRO ID 009, TRACE STOP  TTPRIME  002

   TESTING                    TESTRAN OUTPUT                DATE 66/084              TIME 00/00          PAGE    2
(9)
1) MACRO ID 010, DUMP PANEL
G'09' +1073790202 G'10' -2 G'11' +130932
PSW  FF 0 5 0026 5 0 C057EC  CC=1  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

(10)
AT LOCATION TTSVC2  (CALLTEST) 0CCCEA 00582A  ENTER  DATAGEN

(11)
1) MACRO ID 010, DUMP DATA   STARTING IN SECTION CALLTEST
  00F0     TESTVAR  ANS
005830     -2       -838773242

(12)
EXECUTED STATEMENTS DATAGEN  011, 013

(13)
1) MACRO ID 013, DUMP PANEL
G'15' CCCCCCOO
PSW  FF 0 5 0026 5 0 00582C  CC=1  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

(14)
1) MACRO ID 014, DUMP COMMENT
REGISTER 15 SHOULD CONTAIN 0

AT LOCATION (PRIMER  ) 000058 005778  ENTER  TTPRIME

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000058 005778 TO (PRIMER  ) 0000CA 0057EA, STARTED

(15)
1) MACRO ID 003, DUMP CHANGES

 NONE

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000064 005784, CC=1
SVC 26              G'00' 0000003C G'01' 9000581C G'14' 5000582A G'15' 00005778

AT LOCATION (PRIMER  ) 000064 005784  ENTER  TTPRIME

1) MACRO ID 006, DUMP PANEL
G'10' -1
PSW  FF 0 5 0026 5 0 C05786  CC=1  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF
```

```
1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000066 005786 TO (PRIMER  ) 0000CA 0057EA, CC=1
BC   CO F 072        G'15' 00005778

AT LOCATION (PRIMER  ) 0000CA 0057EA  ENTER  TTPRIME

1) MACRO ID 009, TRACE STOP  TTPRIME  002

1) MACRO ID 010, DUMP PANEL
G'09' +1073790202 G'1C' -1 G'11' +130932
PSW  FF 0 5 0026 5 0 0057EC  CC=1  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

AT LOCATION TTSVC2  (CALLTEST) 0CC0EA 00582A  ENTER  DATAGEN

1) MACRO ID 010, DUMP DATA   STARTING IN SECTION CALLTEST
  00F0     TESTVAR  ANS
005830     -1       -838773242

EXECUTED STATEMENTS DATAGEN  011, 013

1) MACRO ID 013, DUMP PANEL
G'15' CC0C0C00
PSW  FF 0 5 0026 5 0 00582C  CC=1  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

1) MACRO ID 014, DUMP COMMENT
REGISTER 15 SHOULD CONTAIN 0

AT LOCATION (PRIMER  ) 0C0058 005778  ENTER  TTPRIME

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000058 005778 TO (PRIMER  ) 0000CA 0057EA, STARTED

1) MACRO ID 003, DUMP CHANGES

 NONE

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000064 005784, CC=1
SVC 26              G'00' 0003003C G'01' 9000581C G'14' 5000582A G'15' 00005778
```



```
AT LOCATION (PRIMER  ) 000064 005784  ENTER  TTPRIME

1) MACRO ID 006, DUMP PANEL
G'1C' +0
PSW  FF 0 5 0026 5 0 005786  CC=1  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000066 005786 TO (PRIMER  ) 0000CA 0057EA, CC=0
BC   CO F 072        G'15' 00005778

AT LOCATION (PRIMER  ) 0000CA 0057EA  ENTER  TTPRIME

1) MACRO ID 009, TRACE STOP  TTPRIME  002

1) MACRO ID 010, DUMP PANEL
G'09' +10737902C2 G'1C' +0 G'11' +130932
PSW  FF 0 5 0026 4 0 0057EC  CC=0  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

AT LOCATION TTSVC2  (CALLTEST) 0CC0EA 00582A  ENTER  DATAGEN

1) MACRO ID 010, DUMP DATA   STARTING IN SECTION CALLTEST
  00F0     TESTVAR  ANS
005830     +0       -838773242

EXECUTED STATEMENTS DATAGEN  011, 013

1) MACRO ID 013, DUMP PANEL
G'15' CC0C0C00
PSW  FF 0 5 0026 4 0 00582C  CC=0  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

1) MACRO ID 014, DUMP COMMENT
REGISTER 15 SHOULD CONTAIN 0

AT LOCATION (PRIMER  ) 0C0058 005778  ENTER  TTPRIME

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000058 005778 TO (PRIMER  ) 0000CA 0057EA, STARTED
```

```
1) MACRO ID 003, DUMP CHANGES

 NONE

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000064 005784, CC=0
SVC 26              G'00' 0000003C G'01' 8000581C G'14' 4000582A G'15' 00005778

AT LOCATION (PRIMER  ) 000064 005784  ENTER  TTPRIME

1) MACRO ID 006, DUMP PANEL
G'10' +1
PSW  FF 0 5 0026 4 0 005786  CC=0  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000072 005792 TO ODD     (PRIMER  ) 000082 0057A2, CC=1
BC   7C F 02A        G'15' 00005778

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000094 005784 TO GOT     (PRIMER  ) 000088 0057D8, CC=2
BC   20 F 060        G'15' 00005778

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 0000C0 0057E0, CC=2
SVC 26              G'0C' C000003C G'01' 8000581C G'14' 4000582A G'15' 00005778

AT LOCATION (PRIMER  ) 0000C0 0057E0  ENTER  TTPRIME

1) MACRO ID 009, TRACE STOP  TTPRIME  002

1) MACRO ID 010, DUMP PANEL
G'09' +3 G'10' +0 G'11' +3
PSW  FF 0 5 0026 6 0 0057E2  CC=2  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

AT LOCATION TTSVC2  (CALLTEST) 0CCCEA 00582A  ENTER  DATAGEN

(16)
1) MACRO ID 010, DUMP DATA   STARTING IN SECTION CALLTEST
  00F0     TESTVAR  ANS
0C5830      +1       +3

(17)
AT LOCATION (PRIMER  ) 000058 005778  ENTER  TTPRIME
```



```
1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000058 005778 TO (PRIMER  ) 0000CA 0057EA, STARTED

1) MACRO ID 003, DUMP CHANGES

 NONE

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000064 005784, CC=2
SVC 26              G'00' C000003C G'01' A000581C G'14' 6000582A G'15' 00005778

AT LOCATION (PRIMER  ) 000064 005784  ENTER  TTPRIME

1) MACRO ID 006, DUMP PANEL
G'10' +2
PSW  FF 0 5 0026 6 0 005786  CC=2  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 00007E 00579E TO LOAD    (PRIMER  ) 00008A 0057AA, CC=2
BC   F0 F 032        G'15' 00005778

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 000094 005784 TO GOT     (PRIMER  ) 0000B8 0057D8, CC=2
BC   2C F 060        G'15' 00005778

1) MACRO ID 002, TRACE FLOW  , TTPRIME , FROM (PRIMER  ) 0000C0 0057E0, CC=2
SVC 26              G'0C' C000003C G'01' A000581C G'14' 6000582A G'15' 00005778

AT LOCATION (PRIMER  ) 0C00C0 0C57E0  ENTER  TTPRIME

1) MACRO ID 009, TRACE STOP  TTPRIME  002

1) MACRO ID 010, DUMP PANEL
G'09' +3 G'10' +0 G'11' +3
PSW  FF 0 5 0026 6 0 0057E2  CC=2  FIX POINT OVERFLOW OFF  DEC OVERFLOW OFF  EXP UNDERFLOW OFF  SIGNIFICANCE OFF

AT LOCATION TTSVC2  (CALLTEST) 0CCCEA 00582A  ENTER  DATAGEN

1) MACRO ID 010, DUMP DATA   STARTING IN SECTION CALLTEST
  00F0     TESTVAR  ANS
0C5830      +2       +3
```

(8) Indicates that a TRACE STOP macro has been encountered. The TRACE that was stopped is in TTPRIME, and has a macro number of 002. (Note that each TESTRAN macro was assigned a number on the assembly listing.)

(9) Output from a DUMP PANEL macro. General registers 9, 10, and 11 are displayed. Note: At this point in the program, general registers 9, 10, and 11 have not been used, and their contents are therefore meaningless. Output from this DUMP PANEL macro becomes meaningful when the subroutine receives valid test data.

(10) Indicates that a TESTRAN SVC has been encountered at location TTSVC2 in control section CALLTEST. The relative address of TTSVC2 is 0000EA, and the loaded address is 00582A.

(11) Output from a DUMP DATA macro. The contents of TESTVAR and ANS are displayed, along with the relative (00F0) and loaded (005830) addresses of TESTVAR. TESTVAR contains -2. Since this data is negative, PRIMER considers it invalid, and, therefore, did not return a value in ANS. The contents of ANS are therefore meaningless on this execution.

(12) Indicates that macros numbered 11 and 13 in control section DATAGEN have been executed. These are the TEST WHEN macro (11) and the DUMP PANEL (G'15') macro. This is an indication that the TEST WHEN macro caused a branch to the DUMP PANEL.

(13) The output of the DUMP PANEL macro.

(14) The output of a DUMP COMMENT macro.

(15) The second execution of DUMP CHANGES indicates that there have been no changes since the last DUMP of the same area.

From this point on, the output follows a pattern similar to that on the first two pages. This is, of course, because the program is looping through the same series of requests for TESTRAN services in the normal course of its execution.

Inspection of the output will reveal the effect of DATAGEN. The value of TESTVAR increases by 1 in each occurrence of the DUMP DATA output. As long as TESTVAR is negative or 0, the subroutine sets register 15 to 0, and DUMPS a comment to that effect.

Eventually, TESTVAR becomes a valid input value to PRIMER. That is, TESTVAR becomes a positive number. The first occurrence of this is on page 5 of the TESTRAN output.

(16) This output line reflects the first valid input to, and output from, PRIMER. TESTVAR contains +1, the input to PRIMER, and ANS contains +3, the next prime integer, as provided by PRIMER.

The logic of PRIMER can be easily checked by referring to the TRACE FLOW output near the middle of the same page. These output lines indicate a branch to ODD, followed by a branch to GOT.

(17) This line indicates that PRIMER has again been entered and a TESTRAN SVC encountered. Because there was valid data supplied, register 15 was not dumped.

The output from this point on is similar to that presented here. In total, 134 pages of output were produced.

APPENDIX: REFERENCE SUMMARY OF TESTRAN MACRO PARAMETERS

The General Form of TESTRAN Macros

Name	Operation	Operand
[any name]	Mnemonic operation code	operand 1, . . . operandn, keyword parameters

Name Field: Identification or entry point

Operation: DUMP, TRACE, TEST, GO, SET

Operand: Positional operands and keyword parameters

Common Keyword Parameters:

SELECT = integer (1 - 8)

Used to control output of TESTRAN editor.

DATAM = data modifier

Specifies attributes to be used.

NAME = any name

Symbolic name to be printed with output

COMMENT = character string

Up to 120 characters of information to be printed with output.

DSECT = dsect name

To identify operands as referring to a dummy control section.

The DUMP Macro

Name	Operation	Operand
[any name]	DUMP	operand 1 [, operand 2] [, operand 3] [, keyword parameters]

Operand 1: DATA, CHANGES, MAP, TABLE, PANEL, or COMMENT

Operand 2: Starting address for DATA and CHANGES; DCB, DEB, or TCB for TABLE; registers to be recorded for PANEL (optional); comment field for COMMENT; not used for MAP.

Operand 3: Ending address for DATA and CHANGES (optional); dcbname for DUMP TABLE, DCB or DUMP TABLE, DEB. Not used for DUMP TABLE, TCB; not used for MAP, PANEL, or COMMENT.

Keyword parameters:

SELECT = integer

DATAM = data modifier

NAME = any name

DSECT = (dsect name [, repeat integer])

The TRACE Macro

Name	Operation	Operand
[any name]	TRACE	Operand 1 [, Operand 2] [, Operand 3] [, keyword parameters]

Operand 1: FLOW, CALL, REFER, or STOP

Operand 2: Starting address for FLOW, CALL, and REFER; name of TRACE macro(s) to be stopped for STOP.

Operand 3: Ending address for FLOW, CALL, or REFER, (optional); not used for STOP.

Keyword Parameters:

SELECT = integer (ALL)

DATAM = data modifier (REFER)

NAME = any name (REFER)

COMMENT = comment field (FLOW, CALL, REFER)

DSECT = (dsect name [, repeat-integer]) (REFER)

The TEST Macro

Name	Operation	Operand
*any name	TEST	OPEN [, operand 2] [, operand 3] [, operand 4] [, keyword parameters]

*mandatory

Operand 2: Specifies return address from execution of OPEN. This is required if OPEN receives control through a problem program branch or if OPEN is entry point of problem program.

Operand 3: Specifies standard page heading.

Operand 4: LINK or LOAD

Keyword parameters:

MAXP = count 1 (specifies the maximum number of pages of test data to be produced)

MAXE = count 2 (specifies the maximum number of test macros to be encountered)

OPTEST = specifies symbolic names of other TEST OPEN macros

SELECT = integer

Name	Operation	Operand
[any name]	TEST	WHEN, operand 2, operand 3, operand 4, operand 5 [, DATAM = Mod]

Operand 2: Value 1 or flagname 1.

Operand 3: Operators AND and OR used with flagname. Comparative operators such as LT and EQ used with values.

Operand 4: Value 2 or flagname 2.

Operand 5: specifies name of next TESTRAN instruction to be executed if test is affirmative.

Name	Operation	Operand
[any name]	TEST	ON, [operand 2, operand 3] , [operand 4, operand 5] [, COUNTER = counter name]

Operand 2: Lower limit

Operand 3: Upper limit

Operand 4: Interval

Operand 5: Specifies name of next TESTRAN instruction to be executed if test is affirmative.

Name	Operation	Operand
[any name]	TEST	AT, (Addr 1, . . . Addrn) [, SELECT = integer]

Name	Operation	Operand
[any name]	TEST	DEFINE, operand 2, (operand 3, . . . operand 3)

Operand 2: Specifies whether a flag or one or more counters are to be defined.

Operand 3: Specifies unique name for each counter or flag, each of which is set to an initial value of 0. Each name is 1 to 8 characters in length. The first character must be a letter.

Name	Operation	Operand
[any name]	TEST	CLOSE

The GO Macro

Name	Operation	Operand
[any name]	GO	operand 1 [, operand 2]

Operand 1: TO, IN, OUT, BACK

Operand 2: Specifies name of next TESTRAN macro to be executed for TO, IN; optional return address for BACK; not used for OUT.

The SET Macro

Name	Operation	Operand
[any name]	SET	operand 1, operand 2, operand 3 [, keyword parameter]

Operand 1: FLAG, COUNTER, VARIABLE

Operand 2: Name for FLAG and COUNTER; symbolic address for VARIABLE

Operand 3: Flagname 2 or condition for FLAG, value for COUNTER and VARIABLE (must be specified as an address)

Keyword Parameter:

DATAM = data modifier for VARIABLE

READER'S COMMENTS

IBM Operating System/360 TESTRAN User's Guide (C20-1652-0)

Your comments regarding this publication will help us improve future editions. Please comment on the usefulness and readability of the publication, suggest additions and deletions, and list specific errors and omissions.

USEFULNESS AND READABILITY

fold fold

SUGGESTED ADDITIONS AND DELETIONS

ERRORS AND OMISSIONS (give page numbers)

fold fold

Name_____

Title or Position_____

Address_____

FOLD ON TWO LINES, STAPLE AND MAIL
No Postage Necessary if Mailed in U.S.A.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Technical Publications

fold fold

IBM
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601

C20-1652-0 Printed in U.S.A.

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601