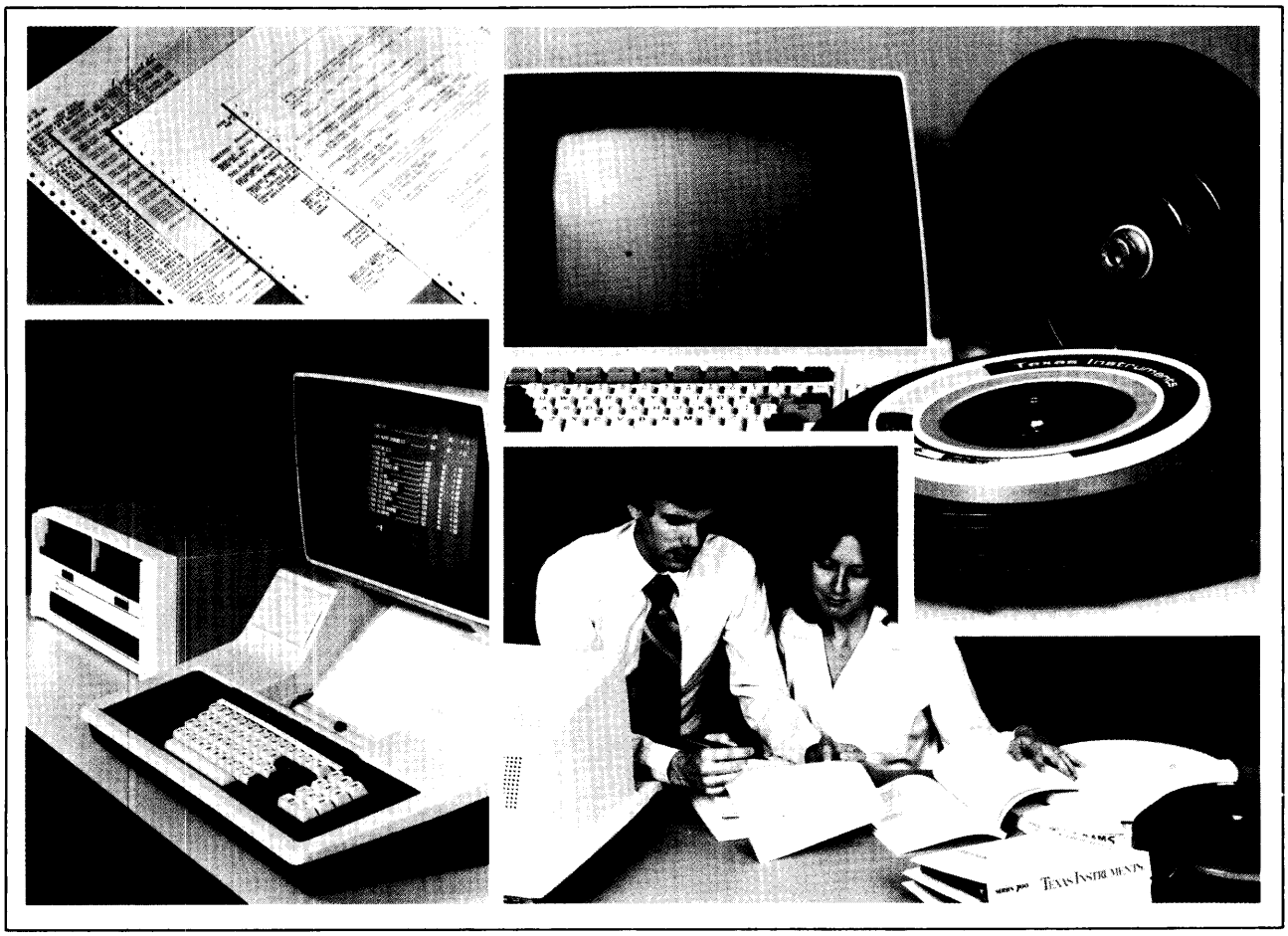
Model 990 Computer
TI Pascal
Configuration Processor Tutorial



Part No. 2250098-9701 *A
1 August 1981

TEXAS INSTRUMENTS

MANUAL REVISION HISTORY

Model 990 Computer TI Pascal Configuration Processor Tutorial
(2250098-9701)

Original Issue . . . . . 15 January 1979
Revision . . . . . 1 August 1981

The total number of pages in this publication is 72.

Preface

This manual is a tutorial about the TI Pascal Configuration Processor. The manual is designed to help you become familiar with the configuration processor, which provides for the separate compilation of TI Pascal (TIP) source modules and configuration management support.

This manual consists of seven sections. Since each section develops information that was presented in the preceding section, you should begin with Section 1 and work through the sections in order.

Section 1 provides an overview of the separate compilation process to give you a basic understanding of how the configuration processor works. This section describes the functions of the configuration processor and other utilities used in this manual; it also lists the steps necessary to compile an entire program or individual routines.

Sections 2 through 7 deal with the actual use of the configuration processor. Each section deals with one phase of the preparation, compilation, linking, and execution of an example program. First, you are told how to create the program by using the Text Editor; then, you are told how to develop this example program step-by-step from creation to execution by using the configuration processor.

Use this manual while sitting at a terminal. The manual directs your activities by first telling you which command to enter and then describing the results. You will need the following equipment:

- DS990 computer system with a DX10 or DNOS operating system and TIP installed
- Model 911 or Model 913 Video Display Terminal (VDT)

After completing this tutorial, you should be able to do the following:

- Use the NESTER utility to format a source program.
- Use the SPLITPGM utility to split a source program into separate modules.
- Use the CONFIG utility to create a hierarchical description (process configuration) of the program.
- Execute the CONFIG utility interactively to prepare one or more source modules for compilation.
- Create a link control file to link the recompiled object modules for execution.

Users of this tutorial should be familiar with SCI commands and the TIP language and should be able to do the following:

- Create a user directory
- Assign a synonym to a directory pathname
- Use the Text Editor to create a source program
- Compile, link, and execute a TIP program

Some commands used in this tutorial display a version number and release date. These are represented as <VERSION: X.X.X YYDDD>, where X.X.X is the version, YY is the year, and DDD is the Julian date.

The following documents contain additional information related to the configuration processor:

Title	Part Number
Model 990 Computer TI Pascal User's Manual	946290-9701
Model 990 Computer DNOS TI Pascal Programmer's Guide	2270517-9701
Model 990 Computer DX10 Operating System Concepts and Facilities Manual	946250-9701
Model 990 Computer DNOS Concepts and Facilities	2270501-9701
Model 990 Computer Link Editor Reference Manual	949617-9701
Model 990 Computer DNOS Link Editor Reference Manual	2270522-9701

You are now ready to begin Section 1.

Contents

Illustrations

Tables

1

Overview

1.1 GENERAL

Development of a large program is less expensive when the modules of the program can be recompiled for correction or change without recompiling the entire program. In a block-structured language such as TIP, separate compilation is more difficult than in assembly language or in a nonstructured, high-level language. This is because of the scope rules of TIP and its ability to pass parameters by reference or by value.

To separately compile a TIP routine, you must include all global declarations in the source code so that the environment is identical to that in which the routine is to execute. These declarations must include the declaration sections of all routines within which the routine is nested. Merging the declaration sections manually is tedious and error prone; in contrast, using the configuration processor is both quicker and more efficient.

1.2 FUNCTIONAL DESCRIPTION OF UTILITIES

You will use the following utilities in the separate compilation process:

- Configuration Processor utility (CONFIG)
- Source Formatter utility (NESTER)
- Split Program utility (SPLITPGM)

CONFIG performs the following functions:

- Maintains a library of source modules to be combined as required for separate compilation of each module of a program
- Builds a process configuration that contains information about the program structure and the locations of the individual modules
- Prepares a source program for each separate compilation, using the process configuration to gather declarations needed by the modules being compiled
- Maintains a library of object modules of the program from which appropriate object modules are linked

NESTER restructures the source code so that the indentation is consistent with the logical structure.

SPLITPGM performs the following:

- Divides a TIP program into modules and catalogs them as members of a directory
- Writes an input command file for the CONFIG utility to contain the commands required to build the process configuration corresponding to the original source program structure

1.3 SEPARATE COMPILATION PROCEDURE

The steps for separate compilation used in this manual are as follows:

1. Prepare the source code (Section 2). Preparation of the source code includes using the Text Editor to write a TIP program, using NESTER to format the source program, and using SPLITPGM to split the source program into individual library members.

2. Execute CONFIG to create a process configuration. You can execute CONFIG in either of two modes: batch or interactive. In the batch mode, CONFIG builds the process configuration automatically, using information supplied in the output file of SPLITPGM. When using the interactive mode, you can enter commands that direct CONFIG in building or modifying the process configuration. In Section 3, you will execute CONFIG in the batch mode. Section 4 discusses the user commands. In Section 5, you will execute CONFIG interactively to prepare the entire program for compilation.

3. Compile, link, and execute the program (Section 6). After compiling and before linking, you will execute CONFIG to split the object code into a separate module for each routine that was compiled.

4. Modify a subroutine and recompile it separately. In Section 7, you will modify one of the subroutines of your example program, recompile only that routine, link it, and execute the program.

Now, proceed to Section 2 to prepare the source code.

2

Preparing the Source Code

2.1 GENERAL

To prepare the source code of the program, first you must create three user directories: one for the source modules, one for the object modules, and one for miscellaneous files such as listings and messages. Next, you must use the Text Editor to create an example program. Then, execute NESTER to format your program. To your nested program, add forward declarations and then mark the start of each routine for SPLITPGM. After executing SPLITPGM you should check the output files.

Some utilities recognize only the first six characters of routine names. Therefore, when preparing source code, use routine names whose first six characters are unique.

The temporary files used by the utilities described in this manual (.COMPFIxx, .CPTEMPxx, .OBJECTxx, etc., where xx is your station number) are automatically deleted when you execute the P$DELETE command or when you log off the system. When the utilities are executed from a batch stream or batch job (DNOS), the name of a temporary directory is prefixed to the default pathname. Synonym $TIP is automatically assigned to this directory. For batch streams initiated by the Execute Batch (XB) command, the temporary directory is .T$STxx, where xx is the station number of the initiating terminal. For batch jobs initiated by the Execute Batch Job (XBJ) command (DNOS), the directory is .T$Jn, where n is the job number (from one to five decimal digits long). In both cases, the temporary directory is automatically deleted when the batch stream ends. To place the files elsewhere, assign synonym $TIP to a directory you have created. However, you must delete the files yourself when finished with them.

2.2 SOURCE FORMAT

You must separate source modules for input to CONFIG and store them as members of a user source library. A source module consists of one program, procedure, or function in which all contained procedures and functions have been replaced by forward declarations. You will use SPLITPGM to divide your example program into source modules in accordance with this rule. In addition, the source modules must conform to the following rules:

- Each procedure and function requires a forward declaration to ensure that each calling sequence is correctly defined.
- Keyword BEGIN of the compound statement that contains the statements of the program, procedure, or function must be in character positions 1 through 5. The component statements must be indented. (Use NESTER to indent the statements.)
- Keyword END of the compound statement that contains the statements of the program, procedure, or function must be in character positions 1, 2, and 3. The component statements must be indented. (Use NESTER to indent the statements.)

- Compiler option NULLBODY should not be specified in any of the source modules.
- Character position 1 should never contain an asterisk (*) except inside a comment.
- Character position 1 should not contain a minus sign (-) unless character position 2 also contains a minus sign.
- A comment in the declaration section that begins in character position 1 must be closed by a brace (}) in character position 72 or an asterisk and a closed parenthesis (*)) in character positions 71 and 72 of the same or succeeding line.

CONFIG recognizes one or more comments in the declaration section of a module. These comments precede the TYPE or VAR declaration and serve as the documentation section of the module. Comments in this section must begin in character position 1 and end in character position 72 of the same or a succeeding line; you can list these comments separately from the source code. Section 4 describes the LISTDOC command that specifies the modules for which the documentation section is to be listed.

2.3 PROCEDURE

You are now ready to prepare the source code by using the following procedure:

1. Create three user directories, using the following pathnames:

.(your name).CONFIG	This directory will contain miscellaneous files. The maximum number of entries for this directory should be 10 (for example, .JANE.CONFIG).
.(your name).CONFIG.SRC	This directory will contain the source modules. The maximum number of entries for this directory should be 14 (for example, .JANE.CONFIG.SRC).
.(your name).CONFIG.OBJ	This directory will contain the object modules. The maximum number of entries for this directory should be 8 (for example, .JANE.CONFIG.OBJ).

2. Use the Text Editor to create the program listed in Figure 2-1 and store it in a file called MAIN under the directory .(your name).CONFIG.SRC.
3. Execute NESTER to nest the source code in accordance with the indentation rules previously described.

   Enter the following:

   ```
   [ ] XNESTER
   ```

```
PROGRAM TCONFIG;
(*************************************************************************
*                                MAIN                                    *
*************************************************************************)
PROCEDURE SUB2; FORWARD;
PROCEDURE SUB1;
(*************************************************************************
*                                SUB1                                    *
*************************************************************************)
BEGIN
WRITELN('HI! FROM SUB1');
SUB2
END;      (* SUB1 *)
PROCEDURE SUB2;
(*************************************************************************
*                                SUB2                                    *
*************************************************************************)
BEGIN
WRITELN('HI! FROM SUB2');
END;      (* SUB2 *)
PROCEDURE SUB3;
(*************************************************************************
*                                SUB3                                    *
*************************************************************************)
PROCEDURE SUB5; FORWARD;
PROCEDURE SUB4;
(*************************************************************************
*                                SUB4                                    *
*************************************************************************)
BEGIN
WRITELN('HI! FROM SUB4');
SUB5
END;      (* SUB4 *)
PROCEDURE SUB5;
(*************************************************************************
*                                SUB5                                    *
*************************************************************************)
BEGIN
WRITELN('HI! FROM SUB5');
END;      (* SUB5 *)
BEGIN     (* SUB3 *)
WRITELN('HI! FROM SUB3');
SUB4;
SUB5
END;      (* SUB3 *)
BEGIN     (* TCONFIG *)
WRITELN('HI! FROM TCONFIG');
SUB1;
SUB2;
SUB3
END.      (* TCONFIG *)
```

Figure 2-1. Example Program Source File

The following prompts appear on your screen; the responses you should enter are shown next to the prompts:

```
EXECUTE PASCAL SRC PROGRAM NESTER <VERSION: X.X.X   YYDDD>
                SOURCE:  .(your name).CONFIG.SRC.MAIN
         NESTED SOURCE:  .(your name).CONFIG.SRC.NMAIN
         ERROR LISTING:  ME (see note)
              MESSAGES:  ME (see note)
                  MODE:  FOREGROUND
```

NOTE

Entering ME for this response causes messages to be displayed on your terminal. You can have the messages put on a file by entering the pathname of a file such as .(your name).CONFIG.MSSG.

After NESTER is executed, the file .(your name).CONFIG.SRC.NMAIN will contain your nested source code. The contents of the file should be the same as in Figure 2-2.

4. Prepare the nested source file for SPLITPGM. Use the Text Editor to add the following to your nested source file:

   a. A forward declaration for each routine.

   b. A marker for the main program and one for each routine. The markers begin in column 1 and appear before the program statement and before each procedure statement. Each marker consists of a double quote, an ampersand, and the routine name (for example, "&SUB1). After completing this step, your file should look like that shown in Figure 2-3.

5. Assign the synonym LIBRARY to your directory. Enter the Assign Synonym (AS) command as follows:

   ```
   [] AS
   ```

   The following prompts appear on your screen; the responses you should enter are shown next to the prompts:

   ```
   ASSIGN SYNONYM VALUE
           SYNONYM:  LIBRARY
             VALUE:  .(your name).CONFIG.SRC
   ```

6. Execute SPLITPGM to divide your source code into separate modules. SPLITPGM catalogs these modules as members of the directory to which you assigned the synonym LIBRARY. SPLITPGM is executed as a Pascal task. Enter the following:

   ```
   [] XPT
   ```

```
PROGRAM TCONFIG;                                                        00000010
(************************************************************************00000020
*                              MAIN                                    *00000030
************************************************************************)00000040
PROCEDURE SUB2; FORWARD;                                                00000050
PROCEDURE SUB1;                                                         00000060
(************************************************************************00000070
*                              SUB1                                    *00000080
************************************************************************)00000090
BEGIN WRITELN('HI! FROM SUB1');                                         00000100
  SUB2                                                                  00000110
END;                                                    (* SUB1 *)      00000120
PROCEDURE SUB2;                                                         00000130
(************************************************************************00000140
*                              SUB2                                    *00000150
************************************************************************)00000160
BEGIN WRITELN('HI! FROM SUB2');                                         00000170
END;                                                    (* SUB2 *)      00000180
PROCEDURE SUB3;                                                         00000190
(************************************************************************00000200
*                              SUB3                                    *00000210
************************************************************************)00000220
PROCEDURE SUB5; FORWARD;                                                00000230
PROCEDURE SUB4;                                                         00000240
(************************************************************************00000250
*                              SUB4                                    *00000260
************************************************************************)00000270
BEGIN WRITELN('HI! FROM SUB4');                                         00000280
  SUB5                                                                  00000290
END;                                                    (* SUB4 *)      00000300
PROCEDURE SUB5;                                                         00000310
(************************************************************************00000320
*                              SUB5                                    *00000330
************************************************************************)00000340
BEGIN WRITELN('HI! FROM SUB5');                                         00000350
END;                                                    (* SUB5 *)      00000360
BEGIN                                                   (* SUB3 *)      00000370
  WRITELN('HI! FROM SUB3'); SUB4;                                       00000380
  SUB5                                                                  00000390
END;                                                    (* SUB3 *)      00000400
BEGIN                                                 (* TCONFIG *)    00000410
  WRITELN('HI! FROM TCONFIG');                                          00000420
  SUB1; SUB2; SUB3                                                      00000430
END.                                                  (* TCONFIG *)    00000440
```

Figure 2-2. Example Program Nested Source File

```
"&TCONFIG
PROGRAM TCONFIG;                                                        00000010
(************************************************************************00000020
*                            MAIN                                      *00000030
************************************************************************)00000040
PROCEDURE SUB1; FORWARD;                                                00000050
PROCEDURE SUB2; FORWARD;
PROCEDURE SUB3; FORWARD;
"&SUB1
PROCEDURE SUB1;                                                         00000060
(************************************************************************00000070
*                            SUB1                                      *00000080
************************************************************************)00000090
BEGIN WRITELN('HI! FROM SUB1');                                         00000100
  SUB2                                                                  00000110
END;                                                  (* SUB1 *)       00000120
"&SUB2
PROCEDURE SUB2;                                                         00000130
(************************************************************************00000140
*                            SUB2                                      *00000150
************************************************************************)00000160
BEGIN WRITELN('HI! FROM SUB2');                                         00000170
END;                                                  (* SUB2 *)       00000180
"&SUB3
PROCEDURE SUB3;                                                         00000190
(************************************************************************00000200
*                            SUB3                                      *00000210
************************************************************************)00000220
PROCEDURE SUB5; FORWARD;                                                00000230
"&SUB4
PROCEDURE SUB4;                                                         00000240
(************************************************************************00000250
*                            SUB4                                      *00000260
************************************************************************)00000270
BEGIN WRITELN('HI! FROM SUB4');                                         00000280
  SUB5                                                                  00000290
END;                                                  (* SUB4 *)       00000300
"&SUB5
PROCEDURE SUB5;                                                         00000310
(************************************************************************00000320
*                            SUB5                                      *00000330
************************************************************************)00000340
BEGIN WRITELN('HI! FROM SUB5');                                         00000350
END;                                                  (* SUB5 *)       00000360
BEGIN                                                 (* SUB3 *)       00000370
  WRITELN('HI! FROM SUB3'); SUB4;                                       00000380
  SUB5                                                                  00000390
END;                                                  (* SUB3 *)       00000400
BEGIN                                               (* TCONFIG *)      00000410
  WRITELN('HI! FROM TCONFIG');                                          00000420
  SUB1; SUB2; SUB3                                                      00000430
END.                                                (* TCONFIG *)      00000440
```

Figure 2-3. Example Program Edited, Nested Source File

The following prompts appear on your screen; the responses you should enter are shown next to the prompts:

```
EXECUTE TI PASCAL TASK
        PROGRAM FILE:  .TIP.PROGRAM
    TASK NAME OR ID:   SPLITPGM
              INPUT:   .(your name).CONFIG.SRC.NMAIN
             OUTPUT:   .(your name).CONFIG.OUTPUT
           MESSAGES:   .(your name).CONFIG.MSSG
    MODE (F, B, D):    F (foreground)
             MEMORY:   leave blank
```

Respond to the prompt PROGRAM FILE by entering the name of the program file where SPLITPGM is stored.

Respond to the prompt OUTPUT by entering the name of the output file that SPLITPGM produces. This file will contain the commands needed to build the process configuration.

7. Execute a List Directory (LD) command to be sure that a separate module for each routine has been stored in your directory (assigned synonym LIBRARY). Enter the command as follows:

```
[] LD
```

The following prompts appear on your screen; the responses you should enter are shown next to the prompts:

```
LIST DIRECTORY
            PATHNAME:  LIBRARY
LISTING ACCESS NAME:   leave blank
```

The following appears on your screen:

```
DIRECTORY LISTING OF:  .JANE.CONFIG.SRC
  MAX # OF ENTRIES: 17     # OF ENTRIES AVAILABLE: 9


 FILE        ALIAS OF  RECORDS     LAST UPDATE        FMT  TYPE    BLK PROTECT
  MAIN          *          52   10/09/80  15:54:12   BS   N SEQ   YES
  NMAIN         *          53   10/09/80  16:04:55   BS   N SEQ   YES
  SUB1          *           8   10/09/80  15:58:15   BS   N SEQ   YES
  SUB2          *           7   10/09/80  15:58:22   BS   N SEQ   YES
  SUB3          *          10   10/09/80  15:58:31   BS   N SEQ   YES
  SUB4          *           8   10/09/80  15:58:30   BS   N SEQ   YES
  SUB5          *           7   10/09/80  15:58:30   BS   N SEQ   YES
  TCONFI        *          12   10/09/80  15:58:33   BS   N SEQ   YES
16:05:51 THURSDAY, OCT 09, 1980.
```

8. Now, execute a Show File (SF) command to look at the output file that SPLITPGM produces. It contains commands that CONFIG will use to build the process configuration. Enter the following:

   [ ]SF

   The following prompts appear on your screen; the response you should enter is shown next to the prompt FILE PATHNAME:

   SHOW FILE
   FILE PATHNAME: .(your name).CONFIG.OUTPUT

   The file should contain the following:

```
*BUILD PROCESS
*ADD TCONFIG
*ADD TCONFIG : SUB1
*ADD TCONFIG : SUB2
*ADD TCONFIG : SUB3
*ADD SUB3    : SUB4
*ADD SUB3    : SUB5
*CAT PROCESS <LIBRARY,PROCESS>
```

You have now finished preparing the source code. Proceed to Section 3.

3

Creating a Process Configuration

3.1 GENERAL

In this section, you will execute CONFIG in the batch mode to create a process configuration. The process configuration is a structural description of your program. CONFIG uses this description to prepare your program for compilation.

CONFIG uses the commands listed in the output file of SPLITPGM to build the process configuration. CONFIG then stores the process configuration in a file called PROCES and catalogs it as a member of your source library (the directory you created and to which you assigned the synonym LIBRARY). Also, CONFIG produces a command listing file that contains a listing of the commands from the SPLITPGM output file, a copy of the process configuration, and a list of the files and synonyms used.

After executing CONFIG, you will execute two SF commands: one to look at the PROCES file and one to look at the command listing file.

3.2 PROCEDURE

Perform each step in the following procedure to create the process configuration:

1. Execute the CONFIG utility. Enter the following command:

   [ ] XCONFIG

   The following prompts appear on your screen; the responses you should enter are shown next to the prompts:

```
EXECUTE CONFIGURATION PROCESSOR <VERSION: X.X.X  YYDDD>
          COMMANDS:  .(your name).CONFIG.OUTPUT
           CRT FILE:  DUMY
           LISTING:  .(your name).CONFIG.LISTING
          MESSAGES:  .(your name).CONFIG.MSSG
              MODE:  BATCH
            SOURCE:  leave blank
            OBJECT:  leave blank
            MEMORY:  4,8
```

   Respond to the prompt COMMANDS by entering the name of the output file that SPLITPGM created. This file contains the commands needed to build the process configuration.

   Respond to the prompt LISTING by entering the name of the command listing file.

In response to the prompt MESSAGES, enter the name of the file to which messages are sent.

Respond to the prompt MEMORY by specifying the stack and heap allocations.

2. Execute an SF command to look at the PROCES file. Enter the following:

```
[] SF
```

The following prompts appear on your screen. The response you should enter is shown next to the prompt FILE PATHNAME.

```
SHOW FILE
      FILE PATHNAME:   LIBRARY.PROCES
```

The displayed file should be similar to that shown in Figure 3-1.

3. Execute another SF command to look at the command listing. Respond to the prompt FILE PATHNAME as shown in the following:

```
[] SF
SHOW FILE
      FILE PATHNAME:   .(your name).CONFIG.LISTING
```

The displayed file should be similar to that shown in Figure 3-2. The file contains a listing of commands SPLITPGM produces, a copy of the process configuration, and a list of the files and synonyms used, in that order. Note that files .COMPFI16, .CPTEMP16, and .OBJECT16 are empty; LIBRARY is the only synonym that has been assigned; and 16 denotes the station from which CONFIG was executed. (Also, note that 16 will be replaced by the station number of your terminal.)

```
VERSION1 00 10/09/80 00 16:10:09 0006 0000 0000 0000 0000 00 00 00 00
         02 PROCESS  00           0001 0001 0000 0001 0000 02 80 00 00
TCONFIG  02 TCONFIG  00           0002 0000 0000 0002 0000 02 80 00 00
SUB1     02 SUB1     00           0000 0003 0001 0000 0003 02 80 00 00
SUB2     02 SUB2     00           0000 0004 0001 0000 0004 02 80 00 00
SUB3     02 SUB3     00           0000 0005 0001 0005 0000 02 80 00 00
SUB4     02 SUB4     00           0000 0006 0004 0000 0006 02 80 00 00
SUB5     02 SUB5     00           0000 0000 0004 0000 0000 02 80 00 00
LIBTBL   00 00000000 00 00000000 0004 0000 0000 0000 0000 00 00 00 00
MASTER   00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
LIBRARY  00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
OBJLIB   00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
ALTOBJ   00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
FLAGTBL  00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
```

Figure 3-1. PROCES File

```
 DXPSCLCP     1.7.0  81.211    TI 990 CONFIGURATION PROCESSOR  10/09/80 16:10:08
*BUILD PROCESS
*ADD TCONFIG
*ADD TCONFIG : SUB1
*ADD TCONFIG : SUB2
*ADD TCONFIG : SUB3
*ADD SUB3    : SUB4
*ADD SUB3    : SUB5
*CAT PROCESS <LIBRARY,PROCESS>


PROCESS NAME         SOURCE LOCATION      OBJECT LOCATION      FLAGS SET
-------------------- -------------------- -------------------- --------------------
TCONFIG              <LIBRARY ,TCONFIG >
  SUB1               <LIBRARY ,SUB1    >
  SUB2               <LIBRARY ,SUB2    >
  SUB3               <LIBRARY ,SUB3    >
    SUB4             <LIBRARY ,SUB4    >
    SUB5             <LIBRARY ,SUB5    >


INPUT    = .JANE.CONFIG.OUTPUT
CRTFIL   = DUMY
OUTPUT   = .JANE.CONFIG.LISTING
COMPFILE = .COMPFI16
CPTEMP   = .CPTEMP16
OBJECT   = .OBJECT16

MASTER   = .MASTER16
LIBRARY  = .JANE.CONFIG.SRC
OBJLIB   = .OBJLIB16
ALTOBJ   = .ALTOBJ16
```

Figure 3-2. Command Listing File

You have now completed Section 3. You have finished preparing your source code, and the process configuration has been created and stored in your source library. Now proceed to Section 4.

4

User Commands

4.1 GENERAL

This section discusses the commands you can use while executing CONFIG interactively. These commands direct the CONFIG to build or modify a process configuration and prepare one or more source modules for compilation. You will not use the example program in this section. You will continue the preparation of the example program in Section 5.

This section discusses each command and then directs you in executing the command so that you can see how it works. Before beginning, you must execute CONFIG interactively.

4.2 EXECUTING CONFIG INTERACTIVELY

To execute CONFIG interactively, enter the following command:

```
[ ] XCONFIGI
```

The following prompts appear on your screen; the responses you should enter are next to the prompts:

```
EXECUTE CONFIG PROCESSOR INTERACTIVELY <VERSION: X.X.X   YYDDD>
                    LISTING:   .(your name).CONFIG.LISTING
                    SOURCE:
                    OBJECT:
                    MEMORY:    4,8
```

Respond to the prompt LISTING by entering the name of the file that will contain the command listing.

The SOURCE and OBJECT prompts request the names of files that will contain the source and object code, respectively, used by CONFIG. The default files .COMPFIxx and .OBJECTxx (xx is your station number) are used when no file name is specified.

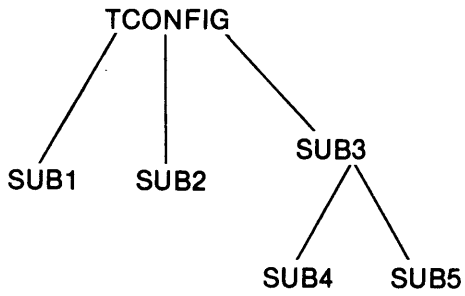
4.3 PROCESS CONFIGURATION

The process configuration contains the following information about the structure of the program:

- Name of the main program
- Names of the routines

- Name of the routine within which each routine is declared (or the main program name for global routines). CONFIG writes, maintains, and uses the process configuration. The user commands specify the structure and contents of the process configuration.

The process configuration is structured as a tree with each node representing a source module of the program. The root node represents the main program module. Your example program is structured as follows:



All of the routines are descendents of the main program. Routines SUB1, SUB2, and SUB3 are sons of TCONFIG; routines SUB4 and SUB5 are sons of SUB3. If another routine were nested in SUB5, it would become the son of SUB5 and a descendent of SUB3.

4.4 COMMAND DESCRIPTION.

Table 4-1 lists and describes the commands available to you. The commands are listed in the order in which they are discussed. Each command is preceded by an asterisk (*) when it is entered. Some of the commands use other special characters, which are explained with the command.

Table 4-1. User Commands

Command	Explanation
USE PROCESS	Specifies which process configuration you intend to use.
DISPLAY	Displays the tabular representation of all or part of the current process configuration.
ADD	Adds a node to the process configuration being built or modified.
MOVE	Moves a node and its descendents to become the son of another node.
DELETE	Deletes a node and its descendents from the current process configuration.
USE OBJECT	Specifies a location for the object module of a specified node.

Table 4-1. User Commands (Continued)

Command	Explanation
USE	Specifies a location for the source module of a specified node.
DEFAULT SOURCE	Specifies a library synonym for the default source library.
DEFAULT OBJECT	Specifies a library synonym for the default object library (the library where the object modules will be stored).
COMPILE	Specifies the source module or modules to be compiled.
EXIT	CONFIG uses this command in the deferred command list to terminate processing. You may enter this command to abort execution of CONFIG without processing the commands entered.
LIST	Lists one or more source modules in the current process configuration.
LISTDOC	Lists the documentation section of one or more source modules in the current process configuration.
LISTORDER	Specifies the listing order for the LIST and LISTDOC commands.
CAT PROCESS	Stores (catalogs) the current process configuration at a specified location.
BUILD PROCESS	Initializes the building of a process configuration.
SETLIB	Defines a library synonym and assigns a value to the synonym.
MASTER	Specifies a library synonym to replace the initially defined synonym MASTER as the first entry in the library table.
LIBRARY	Specifies a library synonym to replace the initially defined synonym LIBRARY as the second entry in the library table.
OBJLIB	Specifies a library synonym to replace the initially defined synonym OBJLIB as the third entry in the library table.
ALTOBJ	Specifies a library synonym to replace the initially defined synonym ALTOBJ as the fourth entry in the library table.
SETFLAG	Defines a user flag or deletes the definition of a user flag.
Flag	Turns certain system flags and all user flags on or off.
Conditional Flag	Allows you to turn the system flags on or off selectively.

4.4.1 USE PROCESS Command

The USE PROCESS command specifies that an existing process configuration is to be used as the current process configuration. Specify the location of the process as a parameter of the command. The following is an example:

```
*USE PROCESS <LIBRARY, PROCES>
```

Note that the location parameter is enclosed in angle brackets (< >) and consists of the synonym for the library and the file name, separated by a comma. The above command specifies the use of the process configuration that is stored in file PROCES in the directory associated with the synonym LIBRARY.

Now enter the command *USE PROCESS <LIBRARY, PROCES>. Only the commands you enter will be displayed on the screen until you enter a DISPLAY command.

4.4.2 DISPLAY Command

The DISPLAY command displays the tabular representation of all or part of the current process configuration. Thus, the DISPLAY command allows you to see the effects of the commands you enter on the current process. You must specify the node if you want only part of the process configuration displayed. For example, the following command displays the configuration of SUB1 and all of its descendents (if it has any):

```
*DISPLAY SUB1 ALL
```

Now display the entire process by omitting the node name from the command, as follows:

```
*DISPLAY ALL
```

The following is displayed, showing each node and the location of its source module:

```
TCONFIG          <LIBRARY, TCONFIG>
  SUB1           <LIBRARY, SUB1>
  SUB2           <LIBRARY, SUB2>
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

4.4.3 ADD Command

The ADD command adds a node to the current process. You must specify both the node to which you are adding a new node and the new node that you are adding; the following is an example:

```
*ADD MAIN:SUB1
```

This example adds the node SUB1 as the son of MAIN. Note that the two nodes are separated by a colon.

Now, add a node to your current process, by entering the following commands:

```
*ADD SUB1:S1A
*DISPLAY ALL
```

The following is displayed, showing S1A as a son of SUB1:

```
TCONFIG              <LIBRARY, TCONFIG>
  SUB1               <LIBRARY, SUB1>
    S1A              <LIBRARY, S1A>
  SUB2               <LIBRARY, SUB2>
  SUB3               <LIBRARY, SUB3>
    SUB4             <LIBRARY, SUB4>
    SUB5             <LIBRARY, SUB5>
```

Note that the source module for the new node does not have to exist at this time.

Now, enter the following commands:

```
*ADD S1A:XYZ
*DISPLAY ALL
```

The following is displayed, showing XYZ as a son of S1A:

```
TCONFIG              <LIBRARY, TCONFIG>
  SUB1               <LIBRARY, SUB1>
    S1A              <LIBRARY, S1A>
      XYZ            <LIBRARY, XYZ>
  SUB2               <LIBRARY, SUB2>
  SUB3               <LIBRARY, SUB3>
    SUB4             <LIBRARY, SUB4>
    SUB5             <LIBRARY, SUB5>
```

4.4.4 MOVE Command

The MOVE command moves a node and all of its descendents to become the son and descendents of another node. You must specify the node you are moving and the node to which you are moving it. The following is an example:

```
*MOVE SUB1 TO SUB2
```

This example moves SUB1 to become the son of SUB2. All descendents of SUB1 (if any) are also moved to become the descendents of SUB2. Note that the second node specified cannot be a descendent of the first node (the node being moved).

Now, enter the following commands:

```
*MOVE S1A TO SUB2
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG              <LIBRARY, TCONFIG>
  SUB1               <LIBRARY, SUB1>
  SUB2               <LIBRARY, SUB2>
    S1A              <LIBRARY, S1A>
      XYZ            <LIBRARY, XYZ>
  SUB3               <LIBRARY, SUB3>
      SUB4           <LIBRARY, SUB4>
      SUB5           <LIBRARY, SUB5>
```

S1A and its son, XYZ, have been moved to become descendents of SUB2.

4.4.5 DELETE Command

The DELETE command deletes a node and its descendents from the current process configuration. You must specify the node to be deleted as a parameter of the command, as in the following example:

```
*DELETE SUB1
```

This example deletes SUB1 and all of its descendents from the current process.

Now, enter the following commands:

```
*DELETE S1A
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG              <LIBRARY, TCONFIG>
  SUB1               <LIBRARY, SUB1>
  SUB2               <LIBRARY, SUB2>
  SUB3               <LIBRARY, SUB3>
    SUB4             <LIBRARY, SUB4>
    SUB5             <LIBRARY, SUB5>
```

S1A and its son, XYZ, have been deleted.

4.4.6 USE OBJECT Command

The USE OBJECT command specifies a location for the object module of a specified node. You must specify the node and the location of its object module as parameters of the command, as in the following example:

```
*USE OBJECT SUB1 <OBJLIB, SUB1>
```

This example stores the object modules for SUBl under the directory that has been assigned the synonym OBJLIB.

Now, enter the following commands:

```
*USE OBJECT SUB2 <OBJLIB, SUB2>
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>
  SUB1           <LIBRARY, SUB1>
  SUB2           <LIBRARY, SUB2>          <OBJLIB, SUB2>
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

Notice that the location of the object module for SUB2 is also shown.

4.4.7 USE Command

The USE command specifies a location for the source module of a specified node. You must specify the node and the location of its source module as parameters of the command, as in the following example:

```
*USE SUB1 <SRCLIB, NODE1A>
```

This example stores the source module for SUB1 in file NODE1A under the directory that has been assigned the synonym SRCLIB.

Now, enter the following commands:

```
*USE SUB1 <OTHRLIB, SUB001>
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>
  SUB1           <OTHRLIB, SUB001>
  SUB2           <LIBRARY, SUB2>          <OBJLIB, SUB2>
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

The source library for the source module of SUB1 has been changed to OTHRLIB. The name under which SUB1 will be cataloged has been changed to SUB001.

Now, enter the following commands:

```
*USE SUB1 <LIBRARY, SUB1>
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>
  SUB1           <LIBRARY, SUB1>
  SUB2           <LIBRARY, SUB2>          <OBJLIB, SUB2>
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

The source library for SUB1 has been changed back to LIBRARY, and the name under which SUB1 will be cataloged has been changed back to SUB1.

4.4.8 DEFAULT SOURCE Command

The DEFAULT SOURCE command specifies the synonym for the default source library. The default source library synonym is LIBRARY until this command is entered. The synonym specified in this command will apply to all modules defined by subsequent ADD commands. The following is an example of the DEFAULT SOURCE command:

```
*DEFAULT SOURCE SRCLIB
```

This example specifies that the directory associated with synonym SRCLIB will be the default source library. When a new node is added with an ADD command, its source module will be in a file cataloged in the directory associated with synonym SRCLIB.

Now, enter the following commands:

```
*DEFAULT SOURCE SRCLIB
*ADD SUB1:S1A
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>
  SUB1           <LIBRARY, SUB1>
    S1A          <SRCLIB, S1A>
  SUB2           <LIBRARY, SUB2>          <OBJLIB, SUB2>
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

4.4.9 DEFAULT OBJECT Command

The DEFAULT OBJECT command specifies a synonym for the default object library. This synonym is associated with the directory in which the object modules will be stored. The synonym specified in this command will apply to all modules defined by subsequent ADD commands. The following is an example of this command:

```
*DEFAULT OBJECT OBJLI
```

This example specifies that the directory associated with the synonym OBJLI will be the default object library. When a node is added with an ADD command, its object module will be in a file cataloged in the directory associated with synonym OBJLI.

Now, enter the following commands:

```
*DEFAULT OBJECT OBJLI
*ADD SUB1:S1B
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>
  SUB1           <LIBRARY, SUB1>
    S1A          <SRCLIB, S1A>
    S1B          <SRCLIB, S1B>          <OBJLI, S1B>
  SUB2           <LIBRARY, SUB2>        <OBJLIB, SUB2>
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

Synonym ALTOBJ is the default object library synonym until the DEFAULT OBJECT command is used. Synonym OBJLIB is the initially defined alternate object library synonym. However, you can specify any library synonym in the command. Note that the location of the object module is not listed in the process configuration unless the DEFAULT OBJECT or USE OBJECT command is entered.

Now, enter the following commands:

```
*DELETE S1A
*DELETE S1B
```

4.4.10 COMPILE Command

The COMPILE command causes CONFIG to prepare a source file for compilation and specifies the module or modules to be compiled. All source modules to be compiled are put on one file, in the proper order for compilation. You must specify the modules to be compiled as parameters of the command. For example, the following command causes only SUB1 to be compiled:

```
*COMPILE SUB1
```

The following command causes SUB1 and all of its descendents to be compiled:

```
*COMPILE SUB1 ALL
```

You can specify more than one module as follows:

```
*COMPILE SUB1, SUB2
```

The following command specifies that the entire program is to be compiled:

```
*COMPILE ALL
```

The optional keyword NO allows you to inhibit the compilation of a module. For example, the following command specifies that SUB1 is *not* to be compiled:

```
*NO COMPILE SUB1
```

You should consider several guidelines when selecting modules for recompilation. However, since you have not compiled any modules yet, these guidelines will be discussed in a later section. Now, enter the following commands:

```
*COMPILE ALL
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>      0 1
  SUB1           <LIBRARY, SUB1>         0 1
  SUB2           <LIBRARY, SUB2>         0 1
  SUB3           <LIBRARY, SUB3>         0 1
    SUB4         <LIBRARY, SUB4>         0 1
    SUB5         <LIBRARY, SUB5>         0 1
```

Notice that the COMPILE command has set two flags: the declaration flag (0) and the body flag (1). When the declaration flag for a module is set, it indicates that the declaration section for that module is to be included in the modules to be compiled. When the body flag for a module is set, it indicates that the body of that module is to be included in the modules to be compiled. Now, enter the following commands:

```
*NO COMPILE ALL
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>
  SUB1           <LIBRARY, SUB1>
  SUB2           <LIBRARY, SUB2>
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

Notice that both flags have been turned off. Now enter the following commands:

```
*COMPILE SUB2
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>      0
  SUB1           <LIBRARY, SUB1>
  SUB2           <LIBRARY, SUB2>         0 1
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

The declaration flags (0) for TCONFIG and SUB2 have been turned on, indicating that the declaration sections are to be included in the module to be compiled. The body flag (1) for SUB2 has also been turned on. Only the code for SUB2 needs to be included in the module to be compiled. Now, enter the following commands:

```
*NO COMPILE SUB2
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY, TCONFIG>
  SUB1           <LIBRARY, SUB1>
  SUB2           <LIBRARY, SUB2>
  SUB3           <LIBRARY, SUB3>
    SUB4         <LIBRARY, SUB4>
    SUB5         <LIBRARY, SUB5>
```

The declaration flags and the body flag have been turned off.

4.4.11 EXIT Command

You can use the EXIT command to abort the execution of CONFIG without processing the command stream. Now, enter the following command:

```
*EXIT
```

CONFIG terminates; no files are saved. If, on the other hand, you had wanted to save and process the command stream you built, you would press the ENTER key instead of entering the EXIT command.

CONFIG also uses the EXIT command in the deferred command list to terminate processing.

4.4.12 LIST Command

The LIST command causes one or more source modules specified in the current process configuration to be listed in the listing file. You must specify the modules you want listed as parameters of the command. For example, the following command lists the source for SUB1:

```
*LIST SUB1
```

The following command lists the source for SUB1 and all of its descendents:

```
*LIST SUB1 ALL
```

The following command lists the source for SUB1 and SUB2:

```
*LIST SUB1,SUB2
```

The following command lists the entire program:

```
*LIST ALL
```

The optional keyword NO allows you to inhibit the listing of the source for a module. For example, as a result of the following command, the source of SUB2 is not listed:

```
*NO LIST SUB2
```

Since you aborted execution of CONFIG with the EXIT command, you need to reexecute it to continue. Enter the following:

```
[ ] XCONFIGI
```

The following prompts appear; the responses you should enter are shown next to the prompts:

```
EXECUTE CONFIGURATION PROCESSOR INTERACTIVELY <VERSION: X.X.X    YYDDD>
                    LISTING:   .(your name).CONFIG.LISTING
                     SOURCE:
                     OBJECT:
                     MEMORY:   4,8
```

Now, enter the following commands:

```
*USE PROCESS <LIBRARY, PROCES>
*LIST ALL
```

After pressing the RETURN key, press the ENTER key to process the commands. Now, enter an SF command to look at the listing file. Enter the following:

```
[ ] SF
```

The following prompts appear; respond to the prompt FILE PATHNAME as shown:

```
SHOW FILE
                FILE PATHNAME:   .(your name).CONFIG.LISTING
```

The file shown in Figure 4-1 now appears. A listing of all of the source modules has been added in addition to the information usually contained in the listing file. The source modules are listed in the order in which they appeared in the process configuration. Notice that the LIST command sets the list flag (2), indicating that all modules are to be listed.

```
 DXPSCLCP     1.7.0  81.211    TI 990 CONFIGURATION PROCESSOR  10/09/80 16:21:30
*USE PROCESS <LIBRARY, PROCES>
*LIST ALL


PROCESS NAME         SOURCE LOCATION     OBJECT LOCATION     FLAGS SET
-------------------- ------------------- ------------------- -----------------
TCONFIG              <LIBRARY ,TCONFIG >                     2
  SUB1               <LIBRARY ,SUB1    >                     2
  SUB2               <LIBRARY ,SUB2    >                     2
  SUB3               <LIBRARY ,SUB3    >                     2
    SUB4             <LIBRARY ,SUB4    >                     2
    SUB5             <LIBRARY ,SUB5    >                     2


INPUT    = ST16
CRTFIL   = ST16
OUTPUT   = .JANE.CONFIG.LISTING
COMPFILE = .COMPFI16
CPTEMP   = .CPTEMP16
OBJECT   = .OBJECT16

MASTER   = .MASTER16
LIBRARY  = .JANE.CONFIG.SRC
OBJLIB   = .OBJLIB16
ALTOBJ   = .ALTOBJ16

CONFIGURATION PROCESSOR                    10/09/80                    16:22:31
TCONFIG  = .JANE.CONFIG.SRC(TCONFIG )

PROGRAM TCONFIG;                                                        00000010
(************************************************************************00000020
*                              MAIN                                     *00000030
************************************************************************)00000040
PROCEDURE SUB1; FORWARD;                                                00000050
PROCEDURE SUB2; FORWARD;                                                00000060
PROCEDURE SUB3; FORWARD;                                                00000070
BEGIN                                                 (* TCONFIG *)    00000080
  WRITELN('HI! FROM TCONFIG');                                          00000090
  SUB1; SUB2; SUB3                                                      00000100
END.                                                  (* TCONFIG *)    00000110

CONFIGURATION PROCESSOR                    10/09/80                    16:22:32
SUB1     = .JANE.CONFIG.SRC(SUB1     )

PROCEDURE SUB1;                                                         00000010
(************************************************************************00000020
*                              SUB1                                     *00000030
************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB1');                                         00000050
  SUB2                                                                  00000060
END;                                                  (* SUB1 *)       00000070
```

Figure 4-1. Listing File — Source (Sheet 1 of 2)

```
CONFIGURATION PROCESSOR                    10/09/80                    16:22:32
SUB2      = .JANE.CONFIG.SRC(SUB2     )


PROCEDURE SUB2;                                                          00000010
(************************************************************************00000020
*                            SUB2                                       *00000030
************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB2');                                          00000050
END;                                                    (* SUB2 *)      00000060


CONFIGURATION PROCESSOR                    10/09/80                    16:22:32
SUB3      = .JANE.CONFIG.SRC(SUB3     )


PROCEDURE SUB3;                                                          00000010
(************************************************************************00000020
*                            SUB3                                       *00000030
************************************************************************)00000040
PROCEDURE SUB5; FORWARD;                                                 00000050
BEGIN                                                   (* SUB3 *)      00000060
  WRITELN('HI! FROM SUB3'); SUB4;                                        00000070
  SUB5                                                                   00000080
END;                                                    (* SUB3 *)      00000090


CONFIGURATION PROCESSOR                    10/09/80                    16:22:33
SUB4      = .JANE.CONFIG.SRC(SUB4     )


PROCEDURE SUB4;                                                          00000010
(************************************************************************00000020
*                            SUB4                                       *00000030
************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB4');                                          00000050
  SUB5                                                                   00000060
END;                                                    (* SUB4 *)      00000070


CONFIGURATION PROCESSOR                    10/09/80                    16:22:33
SUB5      = .JANE.CONFIG.SRC(SUB5     )


PROCEDURE SUB5;                                                          00000010
(************************************************************************00000020
*                            SUB5                                       *00000030
************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB5');                                          00000050
END;                                                    (* SUB5 *)      00000060
```

Figure 4-1. Listing File — Source (Sheet 2 of 2)

4.4.13 LISTDOC Command

The LISTDOC command causes the documentation section of one or more source modules specified in the current process configuration to be listed in the listing file. You must specify the modules as parameters of the command. For example, the following command lists the documentation section for SUB1:

```
*LISTDOC SUB1
```

The following command lists the documentation for SUB1 and all of its descendents:

```
*LISTDOC SUB1 ALL
```

The following command lists the documentation for SUB1 and SUB2:

```
*LISTDOC SUB1,SUB2
```

The following command lists the documentation for the entire program:

```
*LISTDOC ALL
```

The optional keyword NO allows you to inhibit the listing of the documentation for a module. For example, as a result of the following command, the documentation for SUB2 is not listed:

```
*NO LISTDOC SUB2
```

Since you terminated CONFIG by pressing the ENTER key, you must reexecute it to continue. Enter the following:

```
[ ] XCONFIGI
```

The following prompts appear; the responses you should enter are shown next to the prompts:

```
EXECUTE CONFIGURATION PROCESSOR INTERACTIVELY <VERSION: X.X.X   YYDDD>
                    LISTING:  .(your name).CONFIG.LISTING
                    SOURCE:
                    OBJECT:
                    MEMORY:   4,8
```

Now enter the following commands:

```
*USE PROCESS SOURCE <LIBRARY, PROCES>
*LISTDOC ALL
```

After pressing the last RETURN key, press the ENTER key to process the commands. Now enter a Show File command to look at the listing file. Enter the following:

```
[ ] SF
```

The following prompts appear; respond to the prompt FILE PATHNAME as shown:

```
SHOW FILE
                FILE PATHNAME:   .(your name).CONFIG.LISTING
```

The file shown in Figure 4-2 is displayed. Notice that the LISTDOC command set the LISTDOC flag (3).

4.4.14 LISTORDER Command

The LISTORDER command specifies the listing order for the LIST and LISTDOC commands. The command has two options:

- *LISTORDER ALPHA — Lists the source modules and documentation sections in alphabetic order.
- *LISTORDER PROCESS — Lists the source modules and documentation sections in the order in which they appear in the process configuration.

Note that the LIST and LISTDOC commands list the source modules and documentation sections in the order in which they appear in the current process configuration unless the LISTORDER command is used to specify alphabetic order.

Execute CONFIG by entering the following:

```
[ ] XCONFIGI
```

The following prompts appear; the responses you should enter are shown next to the prompts:

```
EXECUTE CONFIG PROCESSOR INTERACTIVELY <VERSION: X.X.X   YYDDD>
                        LISTING:   .(your name).CONFIG.LISTING
                        SOURCE:
                        OBJECT:
                        MEMORY:   4,8
```

Now, enter the following commands:

```
*USE PROCESS <LIBRARY, PROCES>
*LISTORDER ALPHA
*LIST ALL
```

Press the ENTER key, then enter the following to look at the listing file:

```
[ ] SF
```

The following prompts appear; respond to the prompt FILE PATHNAME as shown:

```
SHOW FILE
                FILE PATHNAME:   .(your name).CONFIG.LISTING
```

The file shown in Figure 4-3 appears.

```
 DXPSCLCP     1.7.0  81.211    TI 990 CONFIGURATION PROCESSOR  10/09/80 16:29:24
*USE PROCESS SOURCE <LIBRARY, PROCES>
*LISTDOC ALL

PROCESS NAME          SOURCE LOCATION      OBJECT LOCATION      FLAGS SET
--------------------- -------------------- -------------------- ------------------
TCONFIG               <LIBRARY ,TCONFIG >                       3
  SUB1                <LIBRARY ,SUB1    >                       3
  SUB2                <LIBRARY ,SUB2    >                       3
  SUB3                <LIBRARY ,SUB3    >                       3
    SUB4              <LIBRARY ,SUB4    >                       3
    SUB5              <LIBRARY ,SUB5    >                       3

INPUT    = ST16
CRTFIL   = ST16
OUTPUT   = .JANE.CONFIG.LISTING
COMPFILE = .COMPFI16
CPTEMP   = .CPTEMP16
OBJECT   = .OBJECT16

MASTER   = .MASTER16
LIBRARY  = .JANE.CONFIG.SRC
OBJLIB   = .OBJLIB16
ALTOBJ   = .ALTOBJ16


TCONFIG  = .JANE.CONFIG.SRC(TCONFIG )
(*************************************************************************00000020
*                              MAIN                                       *00000030
*************************************************************************)00000040


SUB1     = .JANE.CONFIG.SRC(SUB1    )
(*************************************************************************00000020
*                              SUB1                                       *00000030
*************************************************************************)00000040


SUB2     = .JANE.CONFIG.SRC(SUB2    )
(*************************************************************************00000020
*                              SUB2                                       *00000030
*************************************************************************)00000040


SUB3     = .JANE.CONFIG.SRC(SUB3    )
(*************************************************************************00000020
*                              SUB3                                       *00000030
*************************************************************************)00000040


SUB4     = .JANE.CONFIG.SRC(SUB4    )
(*************************************************************************00000020
*                              SUB4                                       *00000030
*************************************************************************)00000040


SUB5     = .JANE.CONFIG.SRC(SUB5    )
(*************************************************************************00000020
*                              SUB5                                       *00000030
*************************************************************************)00000040
```

Figure 4-2. Listing File — Documentation

```
 DXPSCLCP     1.7.0  81.211    TI 990 CONFIGURATION PROCESSOR  10/09/80 16:37:18
*USE PROCESS <LIBRARY, PROCES>
*LISTORDER ALPHA
*LIST ALL


PROCESS NAME          SOURCE LOCATION     OBJECT LOCATION     FLAGS SET
--------------------- ------------------- ------------------- -------------------
TCONFIG               <LIBRARY ,TCONFIG >                     2
  SUB1                <LIBRARY ,SUB1    >                     2
  SUB2                <LIBRARY ,SUB2    >                     2
  SUB3                <LIBRARY ,SUB3    >                     2
    SUB4              <LIBRARY ,SUB4    >                     2
    SUB5              <LIBRARY ,SUB5    >                     2


INPUT    = ST16
CRTFIL   = ST16
OUTPUT   = .JANE.CONFIG.LISTING
COMPFILE = .COMPFI16
CPTEMP   = .CPTEMP16
OBJECT   = .OBJECT16

MASTER   = .MASTER16
LIBRARY  = .JANE.CONFIG.SRC
OBJLIB   = .OBJLIB16
ALTOBJ   = .ALTOBJ16


CONFIGURATION PROCESSOR                  10/09/80                    16:37:48
SUB1     = .JANE.CONFIG.SRC(SUB1    )


PROCEDURE SUB1;                                                      00000010
(*********************************************************************00000020
*                            SUB1                                   *00000030
**********************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB1');                                      00000050
  SUB2                                                               00000060
END;                                                  (* SUB1 *)    00000070


CONFIGURATION PROCESSOR                  10/09/80                    16:37:49
SUB2     = .JANE.CONFIG.SRC(SUB2    )


PROCEDURE SUB2;                                                      00000010
(*********************************************************************00000020
*                            SUB2                                   *00000030
**********************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB2');                                      00000050
END;                                                  (* SUB2 *)    00000060
```

Figure 4-3. Listing File — Alphabetic Order (Sheet 1 of 2)

```
CONFIGURATION PROCESSOR                    10/09/80                    16:37:49
SUB3      = .JANE.CONFIG.SRC(SUB3     )


PROCEDURE SUB3;                                                         00000010
(************************************************************************00000020
*                              SUB3                                     *00000030
************************************************************************)00000040
PROCEDURE SUB5; FORWARD;                                                00000050
BEGIN                                                    (* SUB3 *)    00000060
  WRITELN('HI! FROM SUB3'); SUB4;                                       00000070
  SUB5                                                                  00000080
END;                                                     (* SUB3 *)    00000090


CONFIGURATION PROCESSOR                    10/09/80                    16:37:49
SUB4      = .JANE.CONFIG.SRC(SUB4     )


PROCEDURE SUB4;                                                         00000010
(************************************************************************00000020
*                              SUB4                                     *00000030
************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB4');                                         00000050
  SUB5                                                                  00000060
END;                                                     (* SUB4 *)    00000070


CONFIGURATION PROCESSOR                    10/09/80                    16:37:50
SUB5      = .JANE.CONFIG.SRC(SUB5     )


PROCEDURE SUB5;                                                         00000010
(************************************************************************00000020
*                              SUB5                                     *00000030
************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB5');                                         00000050
END;                                                     (* SUB5 *)    00000060


CONFIGURATION PROCESSOR                    10/09/80                    16:37:50
TCONFIG   = .JANE.CONFIG.SRC(TCONFIG )


PROGRAM TCONFIG;                                                        00000010
(************************************************************************00000020
*                              MAIN                                     *00000030
************************************************************************)00000040
PROCEDURE SUB1; FORWARD;                                                00000050
PROCEDURE SUB2; FORWARD;                                                00000060
PROCEDURE SUB3; FORWARD;                                                00000070
BEGIN                                                    (* TCONFIG *) 00000080
  WRITELN('HI! FROM TCONFIG');                                          00000090
  SUB1; SUB2; SUB3                                                      00000100
END.                                                     (* TCONFIG *) 00000110
```

Figure 4-3. Listing File — Alphabetic Order (Sheet 2 of 2)

4.4.15 CAT PROCESS Command

The CAT PROCESS command causes the current process configuration to be stored (cataloged) at a specified location. You specify the location as a parameter of the command if it is not specified in a BUILD PROCESS command. For example, the following command stores the current process in the PROCES in the directory associated with the synonym LIBRARY:

```
*CAT PROCESS <LIBRARY, PROCES>
```

Execute CONFIG by entering the following:

```
[ ] XCONFIGI
```

The following prompts appear; the responses you should enter are shown next to the prompts:

```
EXECUTE CONFIG PROCESSOR INTERACTIVELY <VERSION: X.X.X   YYDDD>
                        LISTING:   .(your name).CONFIG.LISTING
                        SOURCE:
                        OBJECT:
                        MEMORY:    4,8
```

Enter the following commands:

```
*USE PROCESS <LIBRARY, PROCES>
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG              <LIBRARY ,TCONFIG>
  SUB1               <LIBRARY ,SUB1>
  SUB2               <LIBRARY ,SUB2>
  SUB3               <LIBRARY ,SUB3>
    SUB4             <LIBRARY ,SUB4>
    SUB5             <LIBRARY ,SUB5>
```

You are going to use this process, modify it, and then store it in a new location. Enter the following commands:

```
*ADD SUB2:SUB2A
*CAT PROCESS <LIBRARY, PROCS2>
```

The new process, which includes SUB2A, is stored in a file called PROCS2 in the directory assigned synonym LIBRARY.

Now you will use this new process. Enter the following commands:

```
*USE PROCESS <LIBRARY, PROCS2>
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY ,TCONFIG>
  SUB1           <LIBRARY ,SUB1>
  SUB2           <LIBRARY ,SUB2>
    SUB2A        <LIBRARY ,SUB2A>
  SUB3           <LIBRARY ,SUB3>
    SUB4         <LIBRARY ,SUB4>
    SUB5         <LIBRARY ,SUB5>
```

The original process, PROCES, is still in the directory unmodified. Enter the following commands:

```
*USE PROCESS <LIBRARY, PROCES>
*DISPLAY ALL
```

The following is displayed:

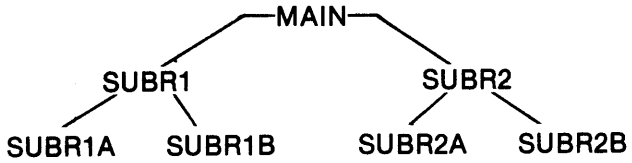
```
TCONFIG          <LIBRARY ,TCONFIG>
  SUB1           <LIBRARY ,SUB1>
  SUB2           <LIBRARY ,SUB2>
  SUB3           <LIBRARY ,SUB3>
    SUB4         <LIBRARY ,SUB4>
    SUB5         <LIBRARY ,SUB5>
```

4.4.16 BUILD PROCESS Command

Until now, you have been using an existing process configuration that was originally built by SPLITPGM. However, you can create a new process with the BUILD PROCESS command, which initializes the building of a process configuration. Specify the location in which your new process is to be stored as a parameter of this command (or specify the location in a CAT PROCESS command). For example, the following command initializes the building of a process that will be stored in a file called NPROCS in the directory associated with the synonym LIBRARY:

```
*BUILD PROCESS <LIBRARY, NPROCS>
```

In the following exercise, you will build a process configuration for the following program structure:



Now, enter the following commands:

```
*BUILD PROCESS <LIBRARY, NPROCS>
*ADD MAIN          (this command specifies MAIN as the root node or main program)
*ADD MAIN:SUBR1
*ADD SUBR1:SUBR1A
*ADD SUBR1:SUBR1B
*ADD MAIN:SUBR2
*ADD SUBR2:SUBR2A
*ADD SUBR2:SUBR2B
*CAT PROCESS
*DISPLAY ALL
```

The following is displayed:

```
MAIN                <LIBRARY ,MAIN>
  SUBR1             <LIBRARY ,SUBR1>
    SUBR1A          <LIBRARY ,SUBR1A>
    SUBR1B          <LIBRARY ,SUBR1B>
  SUBR2             <LIBRARY ,SUBR2>
    SUBR2A          <LIBRARY ,SUBR2A>
    SUBR2B          <LIBRARY ,SUBR2B>
```

Press the ENTER key to process the command and terminate CONFIG.

Three separate process configurations should now be stored in the directory that has been assigned the synonym LIBRARY, as follows:

PROCES — Created by the SPLITPGM utility

PROCS2 — Created by modifying PROCES

NPROCS — Created using the BUILD PROCESS command

Enter a List Directory (LD) command to ensure that these files exist in the directory associated with synonym LIBRARY, as follows:

```
[ ] LD
```

The following prompts appear; respond to the prompt PATHNAME as shown:

```
LIST DIRECTORY
                    PATHNAME:  LIBRARY
        LISTING ACCESS NAME:
```

The following files should be listed:

```
MAIN
NMAIN
NPROCS
PROCES
PROCS2
SUB1
SUB2
SUB3
SUB4
SUB5
TCONFI
```

The remaining paragraphs in this section discuss the library and flag commands. It is not necessary for you to read these paragraphs in order to complete the remaining exercises in this manual. However, you may wish to read them to become familiar with the library and flag commands; if not, proceed to Section 5.

4.4.17 Library Commands

The following paragraphs discuss the commands associated with the library synonyms. A library is identified to CONFIG by using a library synonym. The value of this synonym is the pathname of a library file. To assign a synonym, either use the Assign Synonym (AS) command prior to executing CONFIG (as you did with the synonym LIBRARY) or use the SETLIB command while executing CONFIG.

CONFIG initially defines the following library synonyms:

- MASTER — Intended for source modules of tested (fully developed) programs
- OBJLIB — Intended for object modules corresponding to source modules in MASTER
- LIBRARY — Intended for source modules of programs under development
- ALTOBJ — Intended for object modules corresponding to source modules in LIBRARY

The default source library synonym, LIBRARY, is the logical default because it is intended for programs under development. Similarly, the default object library synonym, ALTOBJ, is appropriate because it is intended for object modules corresponding to the source modules in LIBRARY. You can change either default value using the DEFAULT SOURCE or DEFAULT OBJECT commands.

CONFIG maintains a library table in the process configuration. The first four entries in the table are the initially defined library synonyms MASTER, LIBRARY, OBJLIB, and ALTOBJ. When you enter a library synonym in any of the commands that may include a library synonym parameter, the synonym is added to the library table (unless it already appears in the table). The commands are BUILD PROCESS, CAT PROCESS, USE PROCESS, ADD, USE, USE OBJECT, DEFAULT SOURCE, DEFAULT OBJECT, and EDIT.

You can substitute other synonyms for the initially defined library synonyms. The MASTER command specifies a library synonym to replace MASTER. Similarly, the LIBRARY, OBJLIB, and ALTOBJ commands specify library synonyms to replace LIBRARY, OBJLIB, and ALTOBJ, respectively.

The following paragraphs that discuss the library commands do not require you to enter any commands on the terminal.

4.4.17.1 SETLIB Command. The SETLIB command defines a library synonym and assigns a value to the synonym. Instead of using the AS command before executing CONFIG, you can use SETLIB while executing CONFIG interactively to assign a library synonym such as LIBRARY or ALTOBJ. For example, the following command assigns the value of .JANE.CONFIG.SRC to the synonym SRCLIB:

```
*SETLIB SRCLIB .JANE.CONFIG.SRC
```

4.4.17.2 MASTER Command. The MASTER command specifies a library synonym to replace the initially defined synonym MASTER as the first entry in the library table. When you use the MASTER command, it should precede the ADD commands that define the nodes of the process configuration.

For example, the following command replaces the synonym MASTER with synonym SRCLIB1 as the first entry in the library table:

```
*MASTER SRCLIB1
```

4.4.17.3 LIBRARY Command. The LIBRARY command specifies a library synonym to replace the initially defined synonym LIBRARY as the second entry in the library table. When you use the LIBRARY command, it should precede the ADD commands that define the process configuration.

For example, the following command replaces the synonym LIBRARY with synonym SRCLIB2 as the second entry in the library table:

```
*LIBRARY SRCLIB2
```

4.4.17.4 OBJLIB Command. The OBJLIB command specifies a library synonym to replace the initially defined synonym OBJLIB as the third entry in the library table. When the OBJLIB command is used, it should precede the ADD commands that define the process configuration.

For example, the following command replaces the synonym OBJLIB with synonym OTHLIB as the third entry in the library table:

```
*OBJLIB OTHLIB
```

4.4.17.5 ALTOBJ Command. The ALTOBJ command specifies a library synonym to replace the initially defined synonym ALTOBJ as the fourth entry in the library table. When the ALTOBJ command is used, it should precede the ADD commands that define the process configuration.

For example, the following command replaces the synonym ALTOBJ with synonym OTHLIB2 as the fourth entry in the library table:

```
*ALTOBJ OTHLIB2
```

4.4.18 Flag Commands

The following paragraphs discuss the commands associated with the various flags. The process configuration contains a set of flags for each node; these flags control the processing of the nodes. Each flag is either on or off. Flags are turned on or off by commands.

When all commands have been processed, the states of all flags resulting from processing the commands are passed to the deferred processing run of CONFIG in the external representation of the process configuration that follows the USE PROCESS # command in the deferred command list.

The two categories of flags are system flags and user flags. System flags are predefined and are set to an initial state when a process configuration is built or accessed. The states of system flags are not stored when the process configuration is stored. Table 4-2 lists and explains the system flags.

Table 4-2. System Flags

Flag Number	Flag Name	Description	Initial Value
0	Declaration	Set when declarations of this module are required in the source file.	Off
1	Body	Set when statements of this module are required in the source file.	Off
2	List	Set when the source module is to be listed.	Off
3	Listdoc	Set when the documentation section of this module is to be listed.	Off
4	Changed	Set when contents of a source module are changed by an edit operation.	Off
5	Nest	Cannot be set or cleared by the user.	Off
6	Split	Set when the object module is to be written as a member of library OBJLIB or ALTOBJ.	On
7	Collect	Set when the object module is to be written on the file specified for the OBJECT prompt.	Off
8	Check	Set when the IDT of the module is to be compared to the name of the node.	On

You should already be acquainted with the first four flags listed in Table 4-2.

The COMPILE command turns the declaration flag on for each module for which the declarations are required in the source file being written. The command turns on both the declaration and body flags for modules being compiled. Similarly, the NO COMPILE command turns off the declaration and body flags.

The LIST command turns on the list flag for modules to be listed, and the NO LIST command turns the list flag off for the specified module or modules. Similarly, the LISTDOC command turns on the listdoc flag and the NO LISTDOC command turns the listdoc flag off.

The remaining system flags are discussed in the following paragraphs and in the *TI Pascal User's Manual* and the *TI Pascal Programmer's Guide*.

You can define up to 21 user flags with the SETFLAG command. User flags are turned on and off by the flag command. The states of user flags are stored when the process configuration is stored. You can use the user flags to mark routines that have something in common.

4.4.18.1 SETFLAG Command. The SETFLAG command defines or deletes a user flag. You must specify the flag name and description as parameters of the command. The flag name consists of from one to eight characters and cannot be a keyword of CONFIG. The flag description is a string of up to 64 characters that describes the flag. The flag description begins with the first nonblank character following the flag name and extends to the first asterisk (*), normally the asterisk that begins the next command. The flag description may contain blanks and serves as a comment to identify the flag. When the flag description is omitted, the definition of that flag is deleted, and the flag is turned off in all nodes in the program.

For example, the following command defines flag NEW, which may now be used to mark new subroutines:

```
*SETFLAG NEW NEW SUBROUTINE
```

4.4.18.2 Flag Command. The flag command turns certain system flags and all user flags on or off. For example, the following command turns on the collect flag for SUB1 and causes CONFIG to include a collect object in the deferred command list following the SPLIT OBJECT command:

```
*COLLECT SUB1
```

The deferred processing run of CONFIG writes the modules for which the collect flag is on in the object file. The object file can then be specified in an INCLUDE command to the Linkage Editor.

As another example, the following command turns off the split flags for all modules; consequently, the deferred processing run of CONFIG will not catalog the object modules:

```
*NO SPLIT ALL
```

4.4.18.3 Conditional Flag Command. The Conditional Flag command allows you to turn the system flags on or off selectively. For example, the following command causes modules marked with the flag NEW to be compiled:

```
*IF NEW THEN COMPILE
```

In the following exercise, you will create and set a user-defined flag. Enter the following command:

```
[ ] XCONFIGI
```

The following prompts appear; the responses you should enter are shown next to the prompts:

```
EXECUTE CONFIG PROCESSOR INTERACTIVELY <VERSION: X.X.X   YYDDD>
                        LISTING:   .(your name).CONFIG.LISTING
                        SOURCE:
                        OBJECT:
                        MEMORY:    4,8
```

Now, enter the following commands:

```
*USE PROCESS <LIBRARY, PROCES>
*SETFLAG NEW NEW SUBROUTINE
*NEW SUB1
*NEW SUB2
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY ,TCONFIG>
  SUB1           <LIBRARY ,SUB1>          9
  SUB2           <LIBRARY ,SUB2>          9
  SUB3           <LIBRARY ,SUB3>
    SUB4         <LIBRARY ,SUB4>
    SUB5         <LIBRARY ,SUB5>
```

Notice that flags marking SUB1 and SUB2 are included.

Now, enter the following commands:

```
*IF NEW THEN COMPILE
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY ,TCONFIG>       0
  SUB1           <LIBRARY ,SUB1>          0 1 9
  SUB2           <LIBRARY ,SUB2>          0 1 9
  SUB3           <LIBRARY ,SUB3>
    SUB4         <LIBRARY ,SUB4>
    SUB5         <LIBRARY ,SUB5>
```

Notice that the body flag (1) has been set for SUB1 and SUB2 as well as the declaration flag (0) for TCONFIG, SUB1, and SUB2.

Now, enter the following command:

```
*EXIT
```

This command terminates CONFIG without any processing.

You have now completed Section 4. Now that you are familiar with the user commands, you are ready to prepare your program for compilation. Proceed to Section 5.

5

Preparing the Entire Program for Compilation

5.1 GENERAL

In this section, you will prepare the source code for compilation. First, you will execute CONFIG interactively. Next, you will assign a synonym for the object library and specify the process configuration you wish to use. After adding a couple of commands to the current process, you will process the commands. Then, you will look at the listing file and two other files prepared by CONFIG. One file contains a copy of the process configuration and a list of deferred commands. CONFIG uses these commands in the next run (after compilation). The other file contains the prepared source, which is ready to be compiled.

5.2 PROCEDURE

Prepare your source for compilation by performing the following steps:

1. Execute CONFIG interactively. Enter the following:

```
[ ] XCONFIGI
EXECUTE CONFIG PROCESSOR INTERACTIVELY <VERSION: X.X.X   YYDDD>
                LISTING:   .(your name).CONFIG.LISTING
                SOURCE:
                OBJECT:
                MEMORY:   4,8
```

   Since no file name was entered in response to the prompt SOURCE, CONFIG will store the prepared source in a file called .COMPFInn (where nn is the station number of your terminal).

2. Enter the following command:

```
*SETLIB ALTOBJ .(your name).CONFIG.OBJ
```

   This command assigns the synonym ALTOBJ to the directory in which the object modules will be stored. This could have been done using the Assign Synonym (AS) command before executing CONFIG. (The pathname of the object directory cannot contain a synonym.)

3. Enter the following commands:

```
*USE PROCESS <LIBRARY, PROCES>
*COMPILE ALL
*COLLECT ALL
```

The COLLECT ALL command turns on the collect flag for all source modules and causes CONFIG to include a COLLECT OBJECT command in the deferred command list.

4. Now display the current process configuration. Enter the following:

```
*DISPLAY ALL
```

The following should appear:

```
TCONFIG          <LIBRARY ,TCONFIG>        0 1 7
  SUB1           <LIBRARY ,SUB1>           0 1 7
  SUB2           <LIBRARY ,SUB2>           0 1 7
  SUB3           <LIBRARY ,SUB3>           0 1 7
    SUB4         <LIBRARY ,SUB4>           0 1 7
    SUB5         <LIBRARY ,SUB5>           0 1 7
```

Notice that the declaration flag (0), the body flag (1), and the collect flag (7) have been set for all modules.

5. Press the ENTER key to process the commands.

6. Look at the listing file. Enter the following:

```
[] SF
SHOW FILE
      FILE PATHNAME:   .(your name).CONFIG.LISTING
```

The listing file shown in Figure 5-1 appears. Notice that the declaration flag, the body flag, and the collect flag have been set for all modules. Also, the library synonym ALTOBJ has been assigned a directory pathname.

7. Now look at the file containing the process configuration and the deferred command list. The file is named .CPTEMPnn (where nn is the station number of your terminal). Enter the following:

```
[] SF
SHOW FILE
      FILE PATHNAME:   .CPTEMPnn
```

```
 DXPSCLCP     1.7.0  81.211     TI 990 CONFIGURATION PROCESSOR  10/09/80 16:45:42
*SETLIB ALTOBJ .JANE.CONFIG.OBJ
*USE PROCESS <LIBRARY, PROCES>
*COMPILE ALL
*COLLECT ALL
*DISPLAY ALL


PROCESS NAME          SOURCE LOCATION      OBJECT LOCATION      FLAGS SET
--------------------- -------------------- -------------------- ----------
TCONFIG               <LIBRARY ,TCONFIG >                       0 1 7
  SUB1                <LIBRARY ,SUB1    >                       0 1 7
  SUB2                <LIBRARY ,SUB2    >                       0 1 7
  SUB3                <LIBRARY ,SUB3    >                       0 1 7
    SUB4              <LIBRARY ,SUB4    >                       0 1 7
    SUB5              <LIBRARY ,SUB5    >                       0 1 7


INPUT    = ST16
CRTFIL   = ST16
OUTPUT   = .JANE.CONFIG.LISTING
COMPFILE = .COMPFI16
CPTEMP   = .CPTEMP16
OBJECT   = .OBJECT16

MASTER   = .MASTER16
LIBRARY  = .JANE.CONFIG.SRC
OBJLIB   = .OBJLIB16
ALTOBJ   = .JANE.CONFIG.OBJ
```

Figure 5-1. Listing File

The file listed in Figure 5-2 appears. The USE PROCESS # command that appears at the beginning of the file is different from the USE PROCESS user command. The former specifies that the process that immediately follows is to be used.

The deferred commands (SPLIT OBJECT, COLLECT OBJECT, and EXIT) are used on the next run of CONFIG (after compilation). The SPLIT OBJECT command causes CONFIG to split the object code into object modules and catalog them in the directory that has been assigned synonym ALTOBJ.

```
*USE PROCESS #
VERSION1 00 10/09/80 00 16:47:44 0006 0000 0000 0000 0000 00 00 00 00
         02 PROCESS  00           0001 0001 0000 0001 0000 C3 80 00 00
TCONFIG  02 TCONFIG  00           0002 0000 0000 0002 0000 43 80 00 00
SUB1     02 SUB1     00           0000 0003 0001 0000 0003 43 80 00 00
SUB2     02 SUB2     00           0000 0004 0001 0000 0004 43 80 00 00
SUB3     02 SUB3     00           0000 0005 0001 0005 0000 43 80 00 00
SUB4     02 SUB4     00           0000 0006 0004 0000 0006 43 80 00 00
SUB5     02 SUB5     00           0000 0000 0004 0000 0000 43 80 00 00
LIBTBL   00 00000000 00 00000000 0004 0000 0000 0000 0000 00 00 00 00
MASTER   00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
LIBRARY  00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
OBJLIB   00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
ALTOBJ   00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00
FLAGTBL  00 00000000 00 00000000 0000 0000 0000 0000 0000 00 00 00 00

*SPLIT OBJECT
*COLLECT OBJECT
*EXIT
```

Figure 5-2. Prepared Process File

The COLLECT OBJECT command causes CONFIG to collect the object modules for which the collect flag has been set and store them in the object file. This object file is then specified to the Linkage Editor.

The EXIT command terminates CONFIG.

8. Now look at the prepared source file .COMPFInn (where nn is the station number of your terminal). Enter the following:

```
[] SF
SHOW FILE
     FILE PATHNAME:   .COMPFInn
```

The file listed in Figure 5-3 appears. This file is now ready to be compiled.

You are finished preparing your source file. Now you are ready to compile, link, and execute your program. Proceed to Section 6.

```
(*+          TCONFIG
   +           SUB1
   ,           SUB2
   ,           SUB3
    +            SUB4
    ,            SUB5
 ---       *)
PROGRAM TCONFIG;                                                              00000010
(*****************************************************************************00000020
*                                MAIN                                         *00000030
*****************************************************************************)00000040
PROCEDURE SUB1; FORWARD;                                                      00000050
PROCEDURE SUB2; FORWARD;                                                      00000060
PROCEDURE SUB3; FORWARD;                                                      00000070
(*+                                                                          *)
PROCEDURE SUB1;                                                               00000010
(*****************************************************************************00000020
*                               SUB1                                          *00000030
*****************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB1');                                               00000050
  SUB2                                                                        00000060
END;                                                        (* SUB1 *)        00000070
(*,                                                                          *)
PROCEDURE SUB2;                                                               00000010
(*****************************************************************************00000020
*                               SUB2                                          *00000030
*****************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB2');                                               00000050
END;                                                        (* SUB2 *)        00000060
(*,                                                                          *)
PROCEDURE SUB3;                                                               00000010
(*****************************************************************************00000020
*                               SUB3                                          *00000030
*****************************************************************************)00000040
PROCEDURE SUB5; FORWARD;                                                      00000050
(*+                                                                          *)
PROCEDURE SUB4;                                                               00000010
(*****************************************************************************00000020
*                               SUB4                                          *00000030
*****************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB4');                                               00000050
  SUB5                                                                        00000060
END;                                                        (* SUB4 *)        00000070
(*,                                                                          *)
PROCEDURE SUB5;                                                               00000010
(*****************************************************************************00000020
*                               SUB5                                          *00000030
*****************************************************************************)00000040
BEGIN WRITELN('HI! FROM SUB5');                                               00000050
END;                                                        (* SUB5 *)        00000060
(*-                                                                          *)
BEGIN                                                       (* SUB3 *)        00000060
  WRITELN('HI! FROM SUB3'); SUB4;                                             00000070
  SUB5                                                                        00000080
END;                                                        (* SUB3 *)        00000090
(*-                                                                          *)
BEGIN                                                     (* TCONFIG *)      00000080
  WRITELN('HI! FROM TCONFIG');                                                00000090
  SUB1; SUB2; SUB3                                                            00000100
END.                                                      (* TCONFIG *)      00000110
```

Figure 5-3. Prepared Source File

6

Compiling, Linking, and Executing the Program

6.1 GENERAL

In this section you will compile, link, and execute the entire example program. First, you will execute the TIP compiler to compile the source file and look at the message file to check for compilation errors. Next, you will execute CONFIG for the deferred command run and check the object library to ensure that all object modules have been cataloged. After creating a link control file and program file, you will execute the Linkage Editor. Finally, you will execute your program.

6.2 PROCEDURE

Perform each step in the following procedure:

1. Execute the TIP compiler. Enter the following:

```
[ ] XTIP
EXECUTE TI PASCAL COMPILER <VERSION: X.X.X YYDDD>
                SOURCE:   .COMPFInn
                OBJECT:
                LISTING:  .(your name).CONFIG.LISTING
              MESSAGES:   .(your name).CONFIG.MSSG
               OPTIONS:
                  MEM1:
                  MEM2:
                  MEM3:
```

   Respond to the prompt SOURCE by entering the name of the prepared source file that CONFIG produces. Recall that CONFIG stored the prepared source in file .COMPFInn (where nn is the station number of your terminal).

   You may specify a file for the prompt OBJECT. However, if you do not specify one, the object code will be stored in file .OBJECTnn (where nn is the station number of your terminal).

2. Now, look at the message file produced by the compiler.

   Enter the following:

```
[ ] SF
SHOW FILE
      FILE PATHNAME:   .(your name).CONFIG.MSSG
```

If the compiler executed with no errors, you are ready to proceed to step 3. If you had errors, perform the following steps:

a. Look at the compiler listing file .(your name).CONFIG.LISTING to determine the cause of the errors.

b. Correct the individual source modules stored in the source directory assigned synonym LIBRARY. *Do not* correct the source file that CONFIG prepared (.COMPFInn).

c. Repeat the procedure listed in Section 5 and repeat steps 1 and 2 of this section.

3. Execute CONFIG in the background (batch) mode to process the deferred commands. In this run, CONFIG uses the deferred command list written during its last execution. Recall that commands were stored on file .CPTEMPnn. The deferred commands cause CONFIG to perform the following:

   a. Split the object code into separate modules and store them in the directory you assigned synonym ALTOBJ.

   b. Collect the modules and store them in object file .OBJECTnn. You will specify this object file to the Linkage Editor. This saves you the trouble of writing a separate INCLUDE statement in the link control file for each object module.

   Enter the following:

```
[] XCONFIG
EXECUTE CONFIG PROCESSOR <VERSION: X.X.X   YYDDD>
            COMMANDS:   .CPTEMPnn
              CRT FILE: DUMY
                LISTING: .(your name).CONFIG.LISTING
              MESSAGES: .(your name).CONFIG.MSSG
                  MODE: BACKGROUND
                SOURCE:
                OBJECT: .OBJECTnn
                MEMORY:
```

4. Look at the object library to ensure that object modules have been cataloged in the directory assigned synonym ALTOBJ. Enter the following:

```
[] LD
LIST DIRECTORY
            PATHNAME:   .(your name).CONFIG.OBJ
LISTING ACCESS NAME:
```

The following should appear:

```
DIRECTORY LISTING OF:  .JANE.CONFIG.OBJ
  MAX # OF ENTRIES: 11     # OF ENTRIES AVAILABLE: 5


 FILE        ALIAS OF   RECORDS     LAST UPDATE          FMT   TYPE    BLK PROTECT
  SUB1          *            8    10/09/80   16:56:36    BS    N SEQ   YES
  SUB2          *            8    10/09/80   16:56:37    BS    N SEQ   YES
  SUB3          *            9    10/09/80   16:56:40    BS    N SEQ   YES
  SUB4          *            8    10/09/80   16:56:38    BS    N SEQ   YES
  SUB5          *            8    10/09/80   16:56:39    BS    N SEQ   YES
  TCONFI        *           13    10/09/80   16:56:42    BS    N SEQ   YES
16:57:40 THURSDAY, OCT 09, 1980.
```

5. Create a link control file using the Text Editor. Enter the following in the edit file:

```
NOSYMT
LIBRARY .TIP.OBJ
FORMAT IMAGE, REPLACE, 3
TASK MAIN
INCLUDE (MAIN)
INCLUDE .OBJECTnn
END
```

   Recall that the COLLECT OBJECT command in the deferred command list caused the object modules to be collected on file .OBJECTnn.

   Store this link control file under the pathname .(your name).CONFIG.LC.

6. Execute the Linkage Editor. Enter the following:

```
[] XLE
EXECUTE LINKAGE EDITOR
      CONTROL ACCESS NAME:   .(your name).CONFIG.LC
LINKED OUTPUT ACCESS NAME:   .(your name).CONFIG.PROG
        LISTING ACCESS NAME: .(your name).CONFIG.LINKLIST
                 PRINT WIDTH: 80
```

   If the Linkage Editor executed with no errors and no warnings, you are ready to proceed to step 7. If not, determine the cause of the error and relink.

7. Execute your program. Enter the following:

```
[ ] XPT
EXECUTE TI PASCAL TASK
        PROGRAM FILE:  .(your name).CONFIG.PROG
    TASK NAME OR ID:   MAIN
              INPUT:
             OUTPUT:   LP01
           MESSAGES:   .(your name).CONFIG.MSSG
     MODE (F,B,D):     FOREGROUND
             MEMORY:
```

8. Enter a Show File (SF) command to look at your output. It should appear as follows:

```
HI! FROM TCONFIG
HI! FROM SUB1
HI! FROM SUB2
HI! FROM SUB2
HI! FROM SUB3
HI! FROM SUB4
HI! FROM SUB5
HI! FROM SUB5
```

9. You can print the file if you wish, using the Print File (PF) command.

Now that you have successfully compiled, linked, and executed your program, you are ready to recompile a routine separately. Proceed to Section 7.

7

Recompiling a Routine Separately

7.1 GENERAL

In this section, you will modify the source code of one of the routines in your example program and recompile that routine only. First, you will modify the source module of a routine by using the Text Editor. Next, you will execute CONFIG to prepare that module for compilation. Next, you will execute the compiler and look at the message file. Then, you will execute CONFIG for the deferred command run. Finally, you will link and execute your program.

Keep in mind the following guidelines when selecting modules for recompilation:

- When a statement within the compound statement of a program or routine is changed, recompile the module that contains the program or routine.
- When a declaration of a program is changed (global declaration), recompile the entire program.
- When a declaration of a routine is changed, recompile the module that contains the declaration and the modules of all nodes that are descendents of the node containing the declaration.

7.2 PROCEDURE

Perform each step in the following procedure:

1. Modify SUB1. Recall that the source modules are stored in the directory assigned synonym LIBRARY. Use the Text Editor to modify module SUB1 as follows:

```
PROCEDURE SUB1;
BEGIN WRITELN    ('HI! FROM SUB1');
    SUB2;
    SUB3
  END;
```

2. Execute CONFIG interactively to prepare SUB1 for compilation. Enter the following:

```
[] XCONFIGI
EXECUTE CONFIG PROCESSOR INTERACTIVELY <VERSION: X.X.X   YYDDD>
                LISTING:   .(your name).CONFIG.LISTING
                SOURCE:
                 OBJECT:
                MEMORY:   4,8
```

Now, enter the following commands:

```
*USE PROCESS <LIBRARY, PROCES>
*SETLIB ALTOBJ .(your name).CONFIG.OBJ
*COMPILE SUB1
*DISPLAY ALL
```

The following is displayed:

```
TCONFIG          <LIBRARY ,TCONFIG>        0
  SUB1           <LIBRARY ,SUB1>           01
  SUB2           <LIBRARY ,SUB2>
  SUB3           <LIBRARY ,SUB3>
    SUB4         <LIBRARY ,SUB4>
    SUB5         <LIBRARY ,SUB5>
```

Notice that the compile flag has been set for SUB1; only SUB1 will be recompiled.

Add the following command:

```
*COLLECT ALL
```

This command causes the object modules to be collected on an object file during the next run of CONFIG.

Now, press the ENTER key to process the commands.

3. Execute the TIP compiler. Enter the following:

```
[ ] XTIP
EXECUTE TI PASCAL COMPILER <VERSION: X.X.X   YYDDD>
                SOURCE:   .COMPFInn
                OBJECT:
                LISTING:  .(your name).CONFIG.LISTING
              MESSAGES:   .(your name).CONFIG.MSSG
                OPTIONS:
                   MEM1:
                   MEM2:
                   MEM3:
```

Recall that CONFIG stores the source file in file .CONFInn (where nn is the station number of your terminal). Since no file is specified for the OBJECT: prompt, the object code will be stored in file .OBJECTnn.

4. Now look at the message file. Enter the following:

```
[ ] SF
SHOW FILE
      FILE PATHNAME:   .(your name).CONFIG.MSSG
```

If the compiler executed with no errors, proceed to step 5. Otherwise, determine the cause of the error from the compiler listing file and repeat steps 1 through 4.

5. Execute CONFIG in batch mode to process the deferred commands. During this run, CONFIG performs the following:

   a. Stores the object code for SUB1 in the object library (assigned synonym ALTOBJ)

   b. Collects all object modules and stores them on the object file (.OBJECTnn)

   Enter the following:

```
[ ] XCONFIG
EXECUTE CONFIG PROCESSOR <VERSION: X.X.X   YYDDD>
            COMMANDS:  .CPTEMPnn
              CRT FILE:  DUMY
               LISTING:  .(your name).CONFIG.LISTING
             MESSAGES:  ME
                 MODE:  FOREGROUND
               SOURCE:
               OBJECT:  .OBJECTnn
               MEMORY:  4,8
```

   Respond to the prompt COMMANDS by entering the file containing the deferred command list. Recall that during the last run of CONFIG, the deferred commands were stored in file .CPTEMPnn (where nn is the station number of your terminal).

   At this point, the object code for SUB1 has been stored in the object library and all object modules have been collected and stored on the object file.

6. Execute the Linkage Editor. Since all object modules have been collected and stored on the object file, use the link control file that was created in Section 6. The file should still be stored under the pathname .(your name).CONFIG.LC. You will also use the program file created in Section 6. The program file should be stored under the pathname .(your name).CONFIG.PROG. Enter the following:

```
[ ] XLE
EXECUTE LINKAGE EDITOR
      CONTROL ACCESS NAME:  .(your name).CONFIG.LC
LINKED OUTPUT ACCESS NAME:  .(your name).CONFIG.PROG
        LISTING ACCESS NAME:  .(your name).CONFIG.LINKLIST
                PRINT WIDTH:  80
```

   If the Linkage Editor executed with no errors and no warnings, proceed to step 7. Otherwise, determine the cause of the errors and relink.

7. Execute your program. Enter the following:

```
[ ] XPT
EXECUTE TI PASCAL TASK
        PROGRAM FILE:   .(your name).CONFIG.PROG
    TASK NAME OR ID:    MAIN
              INPUT:
             OUTPUT:    .(your name).CONFIG.OUT
           MESSAGES:    .(your name).CONFIG.MSSG
               MODE:    FOREGROUND
             MEMORY:
```

8. Enter a Show File (SF) command to look at your output. It should appear as follows:

```
HI! FROM TCONFIG
HI! FROM SUB1
HI! FROM SUB2
HI! FROM SUB3
HI! FROM SUB4
HI! FROM SUB5
HI! FROM SUB5
HI! FROM SUB2
HI! FROM SUB3
HI! FROM SUB4
HI! FROM SUB5
HI! FROM SUB5
```

Congratulations, you have now completed the tutorial on the TIP configuration processor.

Alphabetical Index

Introduction

HOW TO USE INDEX

The index, table of contents, list of illustrations, and list of tables are used in conjunction to obtain the location of the desired subject. Once the subject or topic has been located in the index, use the appropriate paragraph number, figure number, or table number to obtain the corresponding page number from the table of contents, list of illustrations, or list of tables.

INDEX ENTRIES

The following index lists key words and concepts from the subject material of the manual together with the area(s) in the manual that supply major coverage of the listed concept. The numbers along the right side of the listing reference the following manual areas:

- Sections — Reference to Sections of the manual appear as "Sections x" with the symbol x representing any numeric quantity.

- Appendixes — Reference to Appendixes of the manual appear as "Appendix y" with the symbol y representing any capital letter.

- Paragraphs — Reference to paragraphs of the manual appear as a series of alphanumeric or numeric characters punctuated with decimal points. Only the first character of the string may be a letter; all subsequent characters are numbers. The first character refers to the section or appendix of the manual in which the paragraph may be found.

- Tables — References to tables in the manual are represented by the capital letter T followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the table). The second character is followed by a dash (-) and a number.

  Tx-yy

- Figures — References to figures in the manual are represented by the capital letter F followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the figure). The second character is followed by a dash (-) and a number.

  Fx-yy

- Other entries in the Index — References to other entries in the index preceded by the word "See" followed by the referenced entry.

USER'S RESPONSE SHEET

Manual Title: Model 990 Computer TI Pascal Configuration Processor Tutorial (2250098-9701)

Manual Date: 1 August 1981 **Date of This Letter:** _____

User's Name: _____ **Telephone:** _____

Company: _____ **Office/Department:** _____

Street Address: _____

City/State/Zip Code: _____

Please list any discrepancy found in this manual by page, paragraph, figure, or table number in the following space. If there are any other suggestions that you wish to make, feel free to include them. Thank you.

Location in Manual	Comment/Suggestion
_____	_____
	_____
	_____
	_____
_____	_____
	_____
	_____
	_____
_____	_____
	_____
	_____
	_____

CUT ALONG LINE

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES (LOCATED ON REVERSE SIDE), TAPE AND MAIL

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 7284 DALLAS, TX

POSTAGE WILL BE PAID BY ADDRESSEE

TEXAS INSTRUMENTS INCORPORATED

DIGITAL SYSTEMS GROUP

ATTN: TECHNICAL PUBLICATIONS
P.O. Box 2909 M/S 2146
Austin, Texas 78769

FOLD

Texas Instruments U.S. District Sales and Service Offices

(A complete listing of U.S. offices is available from the district office nearest your location)

California
831 S. Douglas Street
El Segundo, California 90245
(213) 973-2571

100 California Street
Suite 480
San Francisco, California 94111
(415) 781-9470

776 Palomar Avenue
P.O. Box 9064
Sunnyvale, California 94086
(408) 732-1840*

3186 Airway
Suite J
Costa Mesa, California 92626
(714) 540-7311

Colorado
9725 East Hampden Avenue
Suite 301
Denver, Colorado 80231
(303) 751-1780

Florida
1850 Lee Road
Suite 115
Winter Park, Florida 32789
(305) 644-3535

Georgia
3300 Northeast Expressway
Building 9
Atlanta, Georgia 30341
(404) 458-7791

*Service telephone number

Illinois
515 West Algonquin Road
Arlington Heights, Illinois 60005
(312) 640-2900
(800) 942-0609*

Massachusetts
504 Totten Pond Road
Waltham, Massachusetts 02154
(617) 890-7400

Michigan
24293 Telegraph Road
Southfield, Michigan 48034
(313) 353-0830
(800) 572-8740*

Minnesota
7625 Parklawn Avenue
Minneapolis, Minnesota 55435
(612) 830-1600

Missouri
2368 Schuetz
St. Louis, Missouri 63141
(314) 569-0801*

New Jersey
1245 Westfield Avenue
Clark, New Jersey 07066
(201) 574-9800

Ohio
4124 Linden Avenue
Dayton, Ohio 45432
(513) 258-3877

Pennsylvania
420 Rouser Road
Coraopolis, Pennsylvania 15108
(412) 771-8550

Texas
8001 Stemmons Expressway
P.O. Box 226080
M/S 3108
Dallas, Texas 75266
(214) 689-4460

13510 North Central Expressway
P.O. Box 225214
M/S 393
Dallas, Texas 75265
(214) 238-3881

9000 Southwest Freeway, Suite 400
Houston, Texas 77074
(713) 776-6577

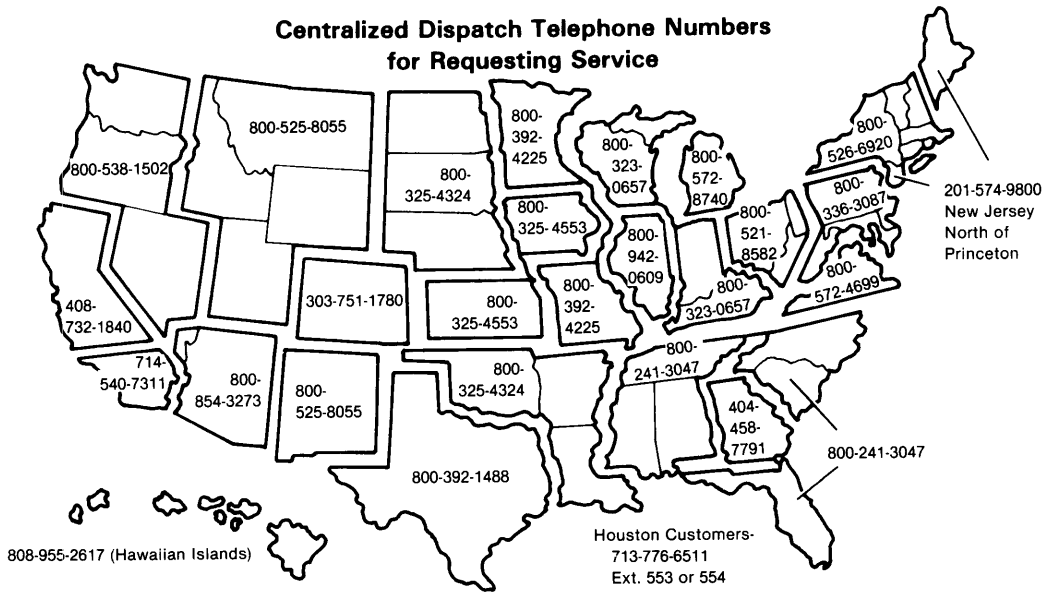
8585 Commerce Drive, Suite 518
Houston, Texas 77036
(713) 776-6531
(713) 776-6553*

Virginia
1745 Jefferson Davis Highway
Crystal Square 4, Suite 600
Arlington, Virginia 22202
(703) 553-2200

Wisconsin
205 Bishops Way
Suite 214
Brookfield, Wisconsin 53005
(414) 784-1323

TI-CARE*

Centralized Dispatch Telephone Numbers for Requesting Service



*Service mark of Texas Instruments

The **TI Customer Support Line** is available to answer our customers' complex technical questions. The extensive experience of a selected group of TI senior engineers and systems analysts is made available directly to our customers. The **TI Customer Support Line** telephone number is **(512) 250-7407.**

TEXAS INSTRUMENTS
INCORPORATED
DIGITAL SYSTEMS GROUP
POST OFFICE BOX 2909 AUSTIN, TEXAS