Manual Update

MANUAL: TI Pascal Configuration Processor Tutorial (2250098-9701)

MCR/CHANGE NO.: MCR 003154/Change 1

EFFECTIVITY DATE: 18 January 1984

This change package contains information necessary to update your current manual. Please remove the obsolete pages from your existing manual and replace them with the changed pages as follows:

Remove Obsolete Pages	Insert Change 1 Pages
Cover/Manual Revision History	Cover/Effective Pages
iii - iv	iii - iv
1-1 - 1-2	1-1 - 1-2
2-3 - 2-8	2-3 - 2-8
3-1 - 3-2	3-1 - 3-2
4-9 - 4-12	4-9 - 4-12
4-15 - 4-16	4-15 - 4-16
4-21 - 4-22	4-21 - 4-22
5-1 - 5-2	5-1 - 5-2
6-3 - 6-4	6-3 - 6-4
7-1 - 7-4	7-1 - 7-4
User's Resp./Bus. Reply	User's Resp./Bus. Reply
Inside Cover/Cover	Inside Cover/Cover

CONFIGURATION PROCESSOR TUTORIAL

TI Pascal

Part No. 2250098-9701 *B
15 January 1984

TEXAS INSTRUMENTS

LIST OF EFFECTIVE PAGES

INSERT LATEST CHANGED PAGES AND DISCARD SUPERSEDED PAGES

Note: The changes in the text are indicated by a change number at the bottom of the page and a vertical bar in the outer margin of the changed page. A change number at the bottom of the page but no change bar indicates either a deletion or a page layout change.

TI Pascal Configuration Processor Tutorial (2250098-9701)

Original issue . . . . . 15 January 1979
Revision . . . . . 1 August 1981
Change 1 . . . . . 15 January 1984

Total number of pages in this publication is 72 consisting of the following:

PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.
Cover	1	4-1 - 4-9	0	6-3 - 6-4	1
Effective Pages	1	4-10 - 4-12	1	7-1	0
iii - iv	1	4-13 - 4-14	0	7-2 - 7-3	1
v - viii	0	4-15 - 4-16	1	7-4	0
1-1 - 1-2	0	4-17 - 4-21	0	Index-1 - Index-4	0
2-1 - 2-3	0	4-22	1	User's Response	1
2-4 -2-6	1	4-23 - 4-28	0	Business Reply	1
2-7	0	5-1	0	Inside Cover	1
2-8	1	5-2	1	Cover	1
3-1	1	5-3 - 5-6	0		
3-2 - 3-4	0	6-1 - 6-2	0		

The computers, as well as the programs that TI has created to use with them, are tools that can help people better manage the information used in their business; but tools—including TI computers—cannot replace sound judgment nor make the manager's business decisions.

Consequently, TI cannot warrant that its systems are suitable for any specific customer application. The manager must rely on judgment of what is best for his or her business.

Preface

This manual is a tutorial about the TI Pascal Configuration Processor. This 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. This 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 operating system or Distributed Network Operating System (DNOS) and TIP installed
- A 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.

Several discussions and instructions in this tutorial refer to the generic key names Enter and Return. See Appendix A of the *TI Pascal Reference Manual*, or the *TI Pascal Programmer's Guide* (either the DNOS or DX10 version), to identify these keys on specific VDTs.

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

Title	Part Number
TI Pascal Reference Manual	946290-9701
DNOS TI Pascal Programmer's Guide	2270517-9701
DX10 TI Pascal Programmer's Guide	2270528-9701
DX10 Operating System Concepts and Facilities	946250-9701
DNOS Concepts and Facilities	2270501-9701
Link Editor Reference Manual	949617-9701
DNOS Link Editor Reference Manual	2270522-9701

You are now ready to begin Section 1.

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 changed 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.

```
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
         NESTER OPTION:  SLIM (0)
         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');                                             00000380
  SUB4;                                                                 00000390
  SUB5                                                                  00000400
END;                                                    (* SUB3 *)      00000410
BEGIN                                                (* TCONFIG *)      00000420
  WRITELN('HI! FROM TCONFIG');                                          00000430
  SUB1;                                                                 00000440
  SUB2;                                                                 00000450
  SUB3                                                                  00000460
END.                                                 (* TCONFIG *)      00000470
```

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');                                             00000380
  SUB4;                                                                 00000390
  SUB5                                                                  00000400
END;                                                    (* SUB3 *)      00000410
BEGIN                                                 (* TCONFIG *)    00000420
  WRITELN('HI! FROM TCONFIG');                                          00000430
  SUB1;                                                                 00000440
  SUB2;                                                                 00000450
  SUB3                                                                  00000460
END.                                                  (* TCONFIG *)    00000470
```

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 : 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 background (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/Background
          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

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 >      <OBJLIB, 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 >      <OBJLIB, 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 >      <OBJLIB, 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 >      <OBJLIB, 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, and 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 and 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.

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 and 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.

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:

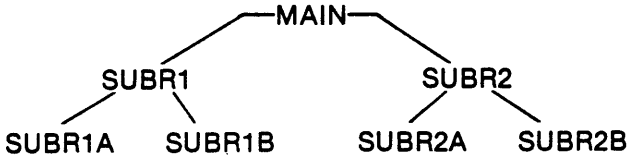
```
TCONFIG          <LIBRARY ,TCONF!G>
  SUB1           <LIBRARY ,SUB1>
  SUB2           <LIBRARY ,SUB2>
  SUB3           <LIBRARY ,SUB3>
    SUB4         <LIBRARY ,SUB4>
    SUB5         <LIBRARY ,SUB5>
```

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:
```

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
```

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 (CHARS):  80
                  PAGE LENGTH:  59
```

   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:     .(your name).CONFIG.OUTPUT
            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:

```
[ ] XCONFIG!
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 (CHARS):  80
                 PAGE LENGTH:  59
```

   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.

USER'S RESPONSE SHEET

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

Manual Date: 15 January 1984 **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
DATA SYSTEMS GROUP

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

FOLD

Cover Part No. 2310002-0001

TEXAS
INSTRUMENTS