APPLICATION		REVISIONS		
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE
	7506			

REV STATUS OF SHEETS																
REV																
SHEET																

UPDATE DOCUMENT, DNOS FORTRAN-78, RELEASE 1.3.0-990

TEXAS INSTRUMENTS INCORPORATED DATA SYSTEMS GROUP	drawing number 2234354-9901		
		REV. **	SHEET 1 OF 30

PREFACE

This document describes the changes to FORTRAN-78 that are relevant to the 1.3.0 release. Numerous enhancements have been implemented in this release and these are described in sufficient detail to allow you to determine what programs you may want to update to take advantage of the enhancements. In general, FORTRAN-78 programs compiled with the 1.2.0 release do not require any source changes to be used with the 1.3.0 release unless you want to use a new feature. Complete re-compilation and re-linking requirements are described in this document.

Approximately 76 STRs were closed with this release of DNOS FORTRAN-78. Known problems that are not closed with this release are described in the DNOS FORTRAN-78 Release Information.

This document is divided into the following three sections:

* Section 1 - describes the important functional changes made for release 1.3.0 and describes the conditions which require re-compiling or re-linking. This section should be carefully reviewed for possible conflicts with your existing applications before installing DNOS FORTRAN-78 on your system.

* Section 2 - describes the new features implemented in this release.

* Section 3 - describes the documentation changes made for release 1.3.0.

* Section 4 - lists the Software Trouble Reports (STRs) that were closed as a result of this release.

Release 1.3.0 supersedes release 1.2.0 of DNOS FORTRAN-78, and replaces all patches for the 1.2.0 release.

SECTION 1

Important Changes

This section describes the important differences between this release of FORTRAN-78 and previous releases. The paragraphs in this section should be carefully reviewed before installing FORTRAN-78 1.3.0 on your system.

1.1 Run-time Uses S$PARM and Task Parameters

In FORTRAN programs compiled with previous releases of FORTRAN-78, the run-time made no use of the PARMS keyword on the .BID, .QBID, or .DBID primitives and did not use PARM1 or PARM2 on the XT, XTS or XHT commands. In FORTRAN-78 1.3.0, the run-time uses the PARMS keyword to pass the pathnames of the run-time error listing file and the default input and output pathnames. The XFT and XFTF commands define these PARMS.

WARNING

If you have FORTRAN application programs which access the first three parameters of the PARMS keyword you must modify the program so that the parameters you use are in the fourth and subsequent parameter positions. The FORTRAN-78 run-time uses the first parameter position for the access name of the error listing file and will write messages to the file specified by this parameter. This could result in irretrievable loss of data in your file should you use a command procedure which provides a data file as the first parameter.

The PARMS list is used by the run-time as follows:

PARM Position	Use
1	Error file or device name
2	Default input unit
3	Default output unit

For tasks linked with the LUNO I/O environment, PARM1 an PARM2 are now used to pass the LUNOs assigned for the default input and output files respectively. These changes may conflict with previous use if the program includes a user-written assembly language subprogram which calls S$PARM to get task bid parameters or issues a Get Task Parameters SVC to access PARM1 or PARM2. Refer to the new features below for more information on how the run-time uses these parameters.

1.2 Change in Logical Name Assignment

The previous release of DNOS FORTRAN-78 assigned the logical name S$FORTRN to the system program file .S$LANG for use with the XF78 and XF78F commands. The 1.3.0 release however, assigns the logical name S$FORTRN to the pathname of the run-time library directory, <volume>.FORT78. The XF78 and XF78F commands now refer to the program file .S$LANG directly rather than through a logical name. This change makes it more convenient to place the run-time object on a secondary disk, since the run-time can be accessed with the logical name.

The installation procedure also assigns the logical name S$MSTAT to the pathname of the MATHSTAT-78 library directory, <volume>.MSTAT78. This logical name assignment is new in the 1.3.0 release.

Both S$FORTRN and S$MSTAT are global logical names.

1.3 ISA Normal Completion Code Changed

In previous releases of FORTRAN-78, the normal completion code for the ISA subroutines was 0. Since this conflicted with the ISA Standards which require the normal completion code to be 1, FORTRAN-78 1.3.0 now uses 1 to indicate normal completion of an ISA extension subprogram. This also provides compatibility with FORTRAN-66 which correctly used 1 as the normal completion code.

1.4 ACCEPT Terminates Field When Full

In the previous release of FORTRAN-78, the ACCEPT statement terminated an input field only after the user had pressed the Return or Enter key. This was a change from the 1.1.0 release and was also not compatible with FORTRAN-66, both of which terminated an input field as soon as it was filled without requiring the Return or Enter key. In the 1.3.0 release of FORTRAN-78, ACCEPT will terminate an input field as soon as it is filled by default. However, a new feature allows you to control field termination with the ACCEPT statement. (The new feature is described later).

1.5 OPEN Uses Default Physical Record Size

In FORTRAN-78 1.3.0, when the OPEN statement creates a file, it uses the default physical record size specified for the disk or directory. In previous releases of FORTRAN-78, the OPEN statement always used a physical record size of 288 bytes.

1.6 BIDTSK Subroutine Deleted

The BIDTSK subroutine has been deleted from DNOS FORTRAN-78 1.3.0 since it used SVC opcode >05 which is not supported. Users migrating from DX10 FORTRAN or DX10 FORTRAN-78 will want to use the EXTASK subroutine instead of BIDTSK.

1.7 CHANGES REQUIRING RECOMPILATION

It is necessary to recompile FORTRAN-78 source programs only for the following reasons:

1. The program uses a value of 1 for the normal completion code with the ISA Extension subprograms. The program must be modified to use a value of 0 for these completion codes.

2. The program accesses the first, second, or third parameters specified with the PARMS keyword on the .BID or .QBID primitive. The program must be modified to use the fourth and subsequent parameters instead.

3. The application depends on the ACCEPT statement to terminate an input field only when the Return or Enter key is pressed by the user. The program must be modified to specify FIELD=FULL in each ACCEPT statement. The default in FORTRAN-78 1.3.0 is to terminate an ACCEPT field as soon as it is filled.

Programs compiled with the 1.2.0 FORTRAN-78 compiler which are not affected by the above can be linked with the 1.3.0 run-ti without recompiling. If you are updating from a release of FORTRAN-78 prior to the 1.2.0 release, you should consult the Release Information document for the 1.2.0 release. In general, it is advisable to recompile all source if you are updating from the 1.0.0 or 1.1.0 releases.

1.8 CHANGES REQUIRING RELINKING

Programs that you have compiled and linked with previous versions of FORTRAN-78 do not need to be relinked if you install FORTRAN-78 1.3.0, except under the following circumstances:

1. If FORTRAN-78 run-time routines are linked as a shared procedure and used by other tasks, all tasks using the run-time shared procedure must be relinked if it becomes necessary to relink any task sharing the procedure.

2. If one or more FORTRAN-78 subprograms are linked as a procedure, whether shared or not, and any other procedure or task segment is relinked using the 1.3.0 FORTRAN-78 run-time, then all procedures, overlays, and the task must be relinked.

SECTION 2

New Features

Several new features are available with this release of FORTRAN-78. The new features are summarized with some brief examples in the following paragraphs.

2.1 Record Locking

This release of FORTRAN-78 provides for record locking with direct access (relative record file) I/O. A LOCK specifier has been added to the direct access READ statement. A new statement, UNLOCK, has been provided to unlock previously locked records.

The following is an example of record locking usage:

```
      OPEN(18,FILE='MYFILE.RELREC',ACCESS='DIRECT',FORM='FORMATTED',
     &     RECL=80,USE='SHARED')
      IREC = 3
5     READ (18, 100, REC=IREC, LOCK, IOSTAT=IOS, ERR = 10) I,J,K
100   FORMAT(3I4)
C     Processing for record ...
C
C     Unlock record
      UNLOCK(18, REC=IREC)
C     Other processing ...
C
C     Handle I/O error codes
10    IF( IOS .EQ. 6 ) THEN
C                         Record is locked by another task -
C                         wait a while...
C         CALL DELAY( 20 )
          GOTO 5
      ELSE
C                         Some I/O error has happened
          STOP 'I/O ERROR AT LABEL 5'
      ENDIF
```

The file MYFILE.RELREC is opened with shared access privileges. The READ statement reads record number three and locks it. If the record is locked by another program, the value of IOS on return from the READ is 6 and the DELAY subprogram is called, which presumably delays for some amount of time. The READ is then executed again. In practice, you may want to restrict th number of times the READ is retried. The UNLOCK statement is used to unlock the record since it is not re-written. If a locked record is written, the record is automatically unlocked. Any records that are locked when the program terminates are unlocked automatically by the operating system.

For more information on record locking and the UNLOCK statement, refer to the FORTRAN-78 Reference Manual.

2.2 Default I/O Units

Default I/O units are supported by this release of FORTRAN-78. A default I/O unit is a file or device preconnected for formatted, sequential I/O. Preconnected means the unit is connected before the program is executed. You preconnect default I/O units by specifying their access names on the XFT or XFTF commands or by specifying their LUNOs on the XT or XTS command.

FORTRAN-78 uses two unique default I/O unit numbers, one for READ and another for WRITE. Only READ and WRITE statements may refer to default I/O units. The syntax for a default I/O unit is an asterisk as shown in the following examples:

```
     READ(*, '(I1)') I
     WRITE(*, '('' THE VALUE OF I IS '',I1)' ) I
```

Note that the default unit is not the same for READ and WRITE unless you connect them to the same file or device. The usual rules for formatted sequential I/O apply. Default unit specifiers may not appear in any I/O statement other then READ and WRITE.

For more information on default I/O units, refer to the FORTRAN-78 Reference Manual.

2.3 STOP Statement Enhancements

The STOP statement syntax has been extended to provide an optional program termination message and the ability to set the value of the SCI completion code synonym ($$CC) to a user-defined value. The message "0040 NORMAL PROGRAM COMPLETION" is no longer issued by the run time. If you want a program termination message, you should use the STOP statement to specify one. The use of the new STOP statement is illustrated by the following examples:

```
     STOP 'PROGRAM COMPLETE'

     STOP (MSG='INCORRECT INPUT', CODE= >8000)
```

The first example displays the message "0038 STOP PROGRAM COMPLETE" when the program terminates. The second example displays the message "0038 STOP INCORRECT INPUT" and sets the value of $$CC to >8000 when the program terminates. The default value of $CC is 0, unless the run time has issued error or warning messages. In that case, the default value is >4000 if warnings were issued, and >8000 if errors were issued. Note that if you specify a completion code value on a STOP statement, the specified value is used, overriding the run time default. Thus if a program has caused warning or error messages to be issued, but executes a STOP (CODE=0), the value of $$CC will be set to 0.

Consult the FORTRAN-78 Reference Manual and DNOS FORTRAN-78 Programmer's Guide for more information on the STOP statement extensions and completion code synonym use.

2.4 IOSTAT and ERR Specifiers

All I/O statements now permit an IOSTAT= and ERR= specifier to be used. These specifiers allow you to handle run time I/O errors in your application rather than automatically terminating the program. The IOSTAT= specifier is used to specify an INTEGER*2 variable name which is defined with a code indicating the success or failure of the I/O operation. The value 0 is always returned for a successful operation. Other possible values are described in the FORTRAN-78 Reference Manual. If you use the ERR= specifier, control is transferred to the executable statement whose label is specified as the value of ERR=. If IOSTAT is also specified, the specified variable is defined with the I/O status value.

Refer to the FORTRAN-78 Reference Manual for more information on the use of the IOSTAT= and ERR= specifiers.

2.5 File Position on OPEN

The OPEN statement has been extended to allow you to specify the initial position of a file when it is opened. A new specifier, POSITION= may be used for this purpose. the value of the specifier is a character constant or variable whose value when all trailing blanks have been removed is REWIND or APPEND If REWIND is specified, the file is positioned at its initial point, or rewound. If APPEND is specified, the file is positioned at the last end of file record in the file. A subsequent WRITE statement would then add records to the file, overwriting the previous end of file. Note that if a disk file has more than one end of file record, the positioning is to the last end of file record. If the file is empty, the file is positioned at its initial point and is the same as though the file was opened with REWIND. If the OPEN statement references a device name rather than a file name, the POSITION specifier must not have the value APPEND. The default if POSITION is not specified is REWIND. The following example illustrates the syntax for the POSITION= specifier.

```
     CHARACTER*12 P
     P = 'APPEND'
     OPEN (UNIT=2, POSITION='REWIND')
     OPEN (4,POSITION=P)
```

2.6 New ACCEPT-DISPLAY Features

Several extensions have been made to the ACCEPT and DISPLAY statements. These extensions permit more flexibility in using VDT features with FORTRAN-78 programs.

2.6.1 ACCEPT Statement Changes.

The ACCEPT statement has the following new specifiers (all are optional):

FIELD=fld. fld is a keyword whose value is FULL or RETURN. You can use the FIELD= specifier to cause ACCEPT to terminate an input field as soon as it is filled, or the user enters a field termination character by specifying FIELD=FULL. If you specify FIELD=RETURN, the user must enter a field termination character to cause ACCEPT to accept the field and continue. Meanwhile, the terminal editing keys (arrow keys, backspace, erase field, and so on, may be used to edit the field.

BLINK. If BLINK appears in the ACCEPT statement, the cursor blinks off and on during the execution of the ACCEPT. If BLINK does not appear in the ACCEPT statement, the cursor does not blink.

NOBEEP. If the NOBEEP specifier appears in the ACCEPT statement, the VDT audible alarm (Control-G, or ASCII BEL) is not sent to the terminal. If NOBEEP is not present, the audible alarm is sounded each time the cursor is positioned for a new field.

INTENSITY=inten. inten is a keyword whose value is HIGH or LOW. If INTENSITY is not specified, HIGH is assumed. You can use INTENSITY=LOW to cause data entered by the user during an ACCEPT to be displayed in high or low intensity as you wish. If you specify PROMPT in the ACCEPT statement, the prompt characters displayed are also affected by the INTENSITY specification.

2.6.2 DISPLAY Statement Changes. The DISPLAY statement has one new specifier, BEEP. If BEEP is specified, the terminal audible alarm (CTRL-G or ASCII BEL) is sent to the terminal when the DISPLAY statement is executed. The default is not to send the audible alarm signal.

2.7 Compiler Enhancements

The FORTRAN-78 compiler has been enhanced in several areas in this release. New options are available to control compiler listing output, a preamble is printed at the front of the listing, a separate listing file may be specified for listing diagnostic messages and a summary of errors and warnings issued, and you can control the number of lines printed one each page. Each of these enhancements is described in the following paragraphs.

2.7.1 New Compiler Options. There are two new compiler options in this release of FORTRAN-78. The new options provide more control of the information included in the listing file produced by the compiler.

* The M option causes the compiler to omit the map information normally included in the source listing. If you specify the M option, the compiler omits the scalar, array, common, and equivalence allocation map information, the subprograms called information, the label map, and the statement location map. If the M option is not specified, this information is included in the source listing.

* The N option causes the compiler to omit the source listing. However, statements which cause diagnostics to be issued will still be listed, along with the diagnostic. Map information is also listed unless the M option is also specified.

2.7.2 Listing File Preamble. The compiler now produces preamble as the first page of the source listing. The preamble contains a replica of the prompts for the XF78 SCI command along with your responses to those prompts or their default values if you did not enter responses to the optional prompts. Any error or warning messages issued by the compiler before the source file is read are included in the preamble page.This would include, for example, invalid option warnings.

2.7.3 Error Listing File. You can have the compiler write statements causing diagnostic messages to a separate listing file by providing the access name on the ERROR ACCESS NAME prompt of the XF78 command. The compiler will write all diagnostic messages and the statements which cause them to the error listing file. The compiler will also print the number of errors and warnings for each program unit compiled and the total number of errors and warnings for the entire compilation. This information is also included in the source listing file, regardless of whether an error listing access name is provided.

The error listing is most useful when compiling a large number of program units, since you can quickly determine where errors occurred without looking through the entire source listing.

2.7.4 Lines Per Page Prompt. There is a new prompt for the XF78 command which allows you to specify the number of lines printed on each page of the source listing and error listing. The prompt is PAGE LENGTH and has a default value of 56. You can specify a shorter or longer page length by entering the number of lines to be printed on each page as a response to this prompt. The number of lines per page includes the header line which the compiler prints as the first line of each page.

2.8 Run-time Error Listing File

With this release of FORTRAN-78, you can specify the file or device to be used by a FORTRAN program for printing all run-time messages. (This change affects only programs linked to use the synonym I/O run-time environment. Programs linked to use the LUNO I/O environment will continue to use the system log files for messages as before). In previous releases, the run-time always printed messages in the Terminal Local File (TLF). A new prompt has been added to the XFT command for this purpose, ERROR ACCESS NAME. If you provide an access name in response to this prompt, all run-time error, warning, debug trace, and information messages will be sent to the file or device specified. If a file pathname is given, the file will be created by the run time if it does not exist. If you do not provide a response to ERROR ACCESS NAME, or if the file or device specified cannot be accessed for some reason, the TLF will be used as before. If the TLF is not empty when the FORTRAN program terminates, the TLF is automatically displayed by SCI. The contents of a file specified for ERROR ACCESS NAME are not automatically displayed, however.

2.9 Run-time Message Handling

There are two changes in the handling of run-time messages in this release of FORTRAN-78. First, the message 0040 NORMAL PROGRAM COMPLETION is no longer used (it is still in the message file however, to provide compatibility with programs compiled and linked before the 1.3.0 version was installed). If you want a message to be displayed when a program terminates normally, (that is, it was not terminated due to an error detected by the run-time) you can use the STOP statement to specify one.

Second, if a FORTRAN program terminates due to an error detected by the run-time, a message is displayed giving the reason for the abnormal termination. In previous releases, these messages were only written to the Terminal Local File, which SCI automatically displayed. In this release, abnormal termination messages are sent to the file or device specified on the ERROR ACCESS NAME prompt of the XFT command (or to the Terminal Local File if no error listing file is specified). The abnormal termination message is also passed to SCI using the S$STOP routine. SCI will then display the message as a termination message at the terminal.

2.10 FORTRAN-66 Compatibility Features

Three new features are implemented in this release of FORTRAN-78 which make it easier to migrate from previous implementations of TI FORTRAN based on the 1966 ANSI standard (FORTRAN-66). These are the DEFINE FILE statement, ENCODE and DECODE statements, and a new object library which provides compatibility with some FORTRAN-66 run time calling conventions.

2.10.1 DEFINE FILE Statement. The DEFINE FILE statement is implemented in this release of FORTRAN-78. The DEFINE FILE statement provides compatibility with previous implementations of TI FORTRAN based on the 1966 ANSI standard, and is provided mainly to make migration to FORTRAN-78 easier. When developi new code with FORTRAN-78, the OPEN statement should be used in place of DEFINE FILE. Refer to the FORTRAN-78 Reference Manual for more information on the DEFINE FILE statement.

2.10.2 ENCODE-DECODE Statements. The ENCODE and DECODE statements are implemented in this release of FORTRAN-78. Like the DEFINE FILE statement, these statements are supported to provide compatibility with previous implementations of TI FORTRAN, making migration to FORTRAN-78 easier. When developing new code with FORTRAN-78, you should use READ and WRITE with internal files in place of ENCODE and DECODE. Refer to the FORTRAN-78 Reference Manual for more information on the ENCODE and DECODE statements.

2.10.3 FORTRAN-66 Compatible Run Time Modules. When FORTRAN-78 was first implemented, some of the library subprograms were duplicated from FORTRAN-66, but the definition of the required arguments were changed to use character data type. This release of FORTRAN-78 makes available a new run-time object library, F66OBJ, which provides subroutines which are functionally equivalent to the FORTRAN-78 versions, but which use the FORTRAN-66 argument definitions. This makes migration to FORTRAN-78 easier. When developing new code with FORTRAN-78, you should us the FORTRAN-78 conventions. Note that you cannot mix both th FORTRAN-66 and FORTRAN-78 versions in the same executable program.

2.11 SCI Command Changes

The commands to execute the FORTRAN-78 compiler and execute FORTRAN tasks have been updated in this release. The changes made are completely compatible with previous versions so that you do not have to change existing batch streams or user documentation unless you wish to use one of the new features.

A new command, F$SYN, is provided in this release.

Most of the changes have already been described above. They are summarized here along with additional changes.

2.11.1 XF78 and XF78F Changes. The XF78 and XF78F commands have the following new prompts:

* ERROR ACCESS NAME - the pathname or device name to be

used for listing error and warning messages.

* PAGE LENGTH - The number of lines per page in the compiler source and error listing files.

* EXECUTION MODE - This prompt appears only on the XF78 command. The responses may be F for foreground or B for background. You need only enter a single letter since the value of the response is determined by the first letter entered. F will execute the FORTRAN-78 compiler in foreground. B, which is the default, will execute the FORTRAN-78 compiler in background. XF78F will always execute the FORTRAN-78 compiler in foreground.

2.11.2 XFT and XFTF Changes. The XFT and XFTF commands have the following new prompts:

* ERROR ACCESS NAME - The file pathname or device to be used by the run-time for messages.

* INPUT - The pathname of a sequential file or device to be used as the input unit by formatted sequential READ statements which specify * as the unit.

* OUTPUT - The pathname of a sequential file or device to be used as the output unit by formatted sequential WRITE statements which specify * as the unit.

* EXECUTION MODE - This prompt appears only on the XFT command. The responses may be F for foreground, B for Background, or D for debug. You need only enter a single letter since the value of the response is determined by the first letter entered. F will execute the FORTRAN task in foreground, B will execute the task in background, and D will execute the task in debug mode, allowing you to use the SCI debugger and debug commands as described in the operating system manuals and the FORTRAN-78 Programmer's Guide. The XFTF command always executes a FORTRAN task in foreground.

2.11.3 F$SYN Command. This command is new in the 1.3.0 release. It deletes all synonyms assigned by the XF78, XF78F, XFT, and XFTF commands, thus helping to prevent synonym table overflow. You may want to include F$SYN in your log-off command (Q or M$01).

2.12 New Installation Procedures

The procedures for installing DNOS FORTRAN-78 have been completely revised. The new features of the installation procedure are:

* There is an SCI command provided with the FORTRAN-78 software kit which installs FORTRAN-78 for you.

* The installation command allows you to install FORTRAN-78 on a secondary disk.

* The installation of expanded message files for the compiler and run-time is optional.

* The installation procedure assigns the logical name S$FORTRN to the run-time library directory pathname, <volume>.FORT78, and assigns the logical name S$MSTAT to the MATHSTAT-78 library directory pathname <volume>.MSTAT78.

* An execute-only installation procedure is provided which allows you to install only the parts of FORTRAN-78 needed to execute FORTRAN-78 programs.

* The installation procedure includes an SCI command which tests the installation by compiling, linking, and executing a test program provided with the software kit.

Refer to the DNOS FORTRAN-78 Object Installation manual for more information.

SECTION 3

DOCUMENTATION CHANGES

The manuals provided for FORTRAN-78 have been completely revised with this release. Aside from documenting the software changes mentioned above, the manuals have been reorganized, portions have been rewritten to improve clarity, and new material is included. The highlights of the documentation changes are summarized in the following paragraphs.

3.1 REFERENCE MANUAL

The FORTRAN-78 Reference Manual has been completely revised. Sections have been reorganized and reordered. New material has been added, especially in the section on Input/Output. All the material describing the run-time library routines (except intrinsic functions) has been moved to the Programmer's Guide.

3.2 PROGRAMMER'S GUIDE

The DNOS FORTRAN-78 Programmer's Guide has been completely revised. Sections have been added on debugging and interfacing to TI productivity software. The material on program development, compilation and execution has been expanded. Link Editing is now described in a separate section and contains much new material. All the run-time library routines are described in the Programmer's Guide now, instead of in the reference manual.

SECTION 4

PROBLEMS FIXED

This section describes the STRs that are closed with the 1.3.0 release of DNOS FORTRAN-78.

4.1 STR 12481

The compiler will no longer drop columns 78, 79, and 80 in the listing if the print width is specified as 132.

4.2 STR 12482

The compiler was fixed to correctly recognize when a label is required on an executable statement and issue a diagnostic if the label is missing.

4.3 STR 12483

The compiler will diagnose an IMPLICIT statement that does not precede all other specification statements.

4.4 STR 12484

A call to the library subroutine TRNON that specifies a time that is less than the current system time will cause the task to be started the following day rather than cause an invalid time error code.

4.5 STR 12486

The compiler will now handle listing file print widths of up to 132 characters correctly.

4.6 STR 12488

The compiler now opens the listing and object files with exclusive write access rather than exclusive all. This permits the file to be read or viewed while the compiler is executing.

4.7 STR 12489

If statements such as ELSE, ELSEIF, and ENDIF are labeled, the compiler now correctly diagnoses illegal transfers to those labels.

4.8 STR 12490

The run-time floating point software was corrected so that the double precision module is automatically loaded into the same segment as the floating point interpreter. This prevents various task errors when the program uses overlays and the double precision module is in an overlay while the floating point module is not.

4.9 STR 12491

When the DO-variable is part of the expression for the initial value in a DO statement (for example, DO 10 I = I +5,50) the compiler correctly computes the number of loop iterations.

4.10 STR 12492

The compiler was fixed so that it does not incorrectly optimize code involving ASSIGN statements and assigned GOTOs.

4.11 STR 12493

The compiler was fixed to eliminate a case of incorrect code and a case of omitted code for certain IF statements.

4.12 STR 12494

A duplicate definition of F$RDBL will no longer result when the alternate floating point module .FORT78.ALTOBJ.F$FITP is included in the link edit.

4.13 STR 12497

The ISA extension subprograms were modified to return the value 1 for normal completion as specified in the ISA Standard documents.

4.14 STR 12498

The compiler was fixed so that it correctly scans statements that resemble key words in a DO loop.

4.15 STR 12499

The run-time was fixed to fill the entire output field with asterisks when the value to be output cannot be represented in the field width, the G edit descriptor is used and the magnitude of the item calls for F editing. The run-time used to fill only part of the field with asterisks.

4.16 STR 12500

The compiler will no longer terminate abnormally when more than fifteen occurrences of the same array subscript expression appear in a block. (A block is a sequence of code beginning with a labeled executable statement).

4.17 STR 12501

The run time will no longer permit an existing sequential file to be opened for direct access.

4.18 STR 12502

The run time now issues a warning if the record length (RECL) of a direct access file is given the value zero. The system default record length is used.

4.19 STR 12504

The run time has been fixed so that if a supervisor call (SVC) error occurs and the I/O statement has used the IOSTAT= or ERR= specifiers, the task is not terminated.

4.20 STR 12505

The ISA OPENW routine will now return an error if it is used for a VDT, since OPENW only opens files for unformatted direct access I/O.

4.21 STR 12506

The D, E, F, and G edit descriptors will now print a zero before the decimal point if the internal value is less than 1.0

4.22 STR 12877

A case of incorrect optimization involving a logical assignment statement followed by a logical IF statement has been fixed. The optimization is now done correctly.

4.23 STR 12912

The OPEN statement can be used to open a VDT without causing the run time to go into an infinite loop.

4.24 STR 13057

The debug trace output sometimes had garbled text and missing module names. This problem has been fixed and the debug trace now works correctly.

4.25 STR 13107

When generating code for DO-loop initialization, the compiler no longer generates code to divide by the loop increment when the increment value is one

4.26 STR 13116

If an SVC error is detected during the execution of an I/O statement that uses the IOSTAT= or ERR= specifiers, the SVC error is correctly handled by the run time and no longer results in task termination.

4.27 STR 13138

The compiler correctly diagnoses the use of subscripted variables as the unit specifier in REWIND, ENDFILE, and BACKSPACE statements. An abnormal termination in the code generation phase used to result, with the message "COMPILER BUG COLLAPSE PHASE".

4.28 STR 13153

A problem causing the compiler to terminate abnormally with the message COMPILER BUG COLLAPSE PHASE has been fixed. The problem was that in a relational expression with a character constant or variable as one of the operands and a non-character function reference as the other operand, the compiler was trying to generate code to convert the character operand to the type of the other operand. Since no such conversion is provided for, the compiler terminated abnormally. The problem was fixed by having the SCAN phase issue an error diagnostic for this situation.

4.29 STR 13201

The compiler has been fixed so that it diagnoses an EQUIVALENCE statement that equivalences two elements of the same array. The allocation phase will issue an error message if this is done.

4.30 STR 13202

If a subscripted variable appears as the format specifier in a READ, WRITE, or DISPLAY statement, a diagnostic is issued and the compiler no longer terminates abnormally in the code generation phase.

4.31 STR 13203

The upper limit on the range of the argument for the SIN, COS, and TAN intrinsic functions was lowered to 2**23 since values larger than this produce meaningless results.

4.32 STR 13204

The intrinsic functions ASIN2, ACOS2, DASIN2, and DACOS2 were fixed to correctly determine the larger of the two arguments and to allow the second argument to have a negative value.

4.33 STR 13205

The complex to integer conversion routine was fixed so that it gives correct results when the conversion is to INTEGER*4.

4.34 STR 13206

If overflow occurs during formatted input, the remaining elements of the I/O list are no longer set to zero. The element that produces the overflow is given the maximum value possible (depending on its type) and the run time continues processing the rest of the format group.

4.35 STR 13207

If a function subprogram defines the function result with a character expression, the compiler recognizes this as a definition of the function name and does not issue the "FUNCTION NOT REFERENCED" diagnostic.

4.36 STR 13211

The compiler now optimizes code generation for the situation I = I + J and generates A @J,@I.

4.37 STR 13603

If a character array reference appears without a subscript, the compiler will issue a diagnostic. The compiler used to terminate with a task error 5 in some cases.

4.38 STR 13732

If a formatted output statement creates a logical record that is larger than the run time I/O buffer size, a warning message is output to tell the user that the record has been truncated.

4.39 STR 13767

The Key Indexed File handler (MTLKIF) will allow a synonym as part of the pathname when linked with the synonym I/O environment. A different version of MTLKIF is included in the LUNO I/O environment that does not permit a synonym.

4.40 STR 13804

The DATE and TIME subroutines have been moved to the OSLOBJ library so that unresolved references do not occur when using the ADATE and MDATE routines. These routines had been placed in the ISAOBJ library by mistake.

4.41 STR 13835

The run time will now issue a message warning about truncated records only if a record is actually truncated. The run time used to issue this warning when the file or device was opened, even though no records had yet been read or written.

4.42 STR 13864

The KIF close subroutine XCLOSE returns a correct status code instead of always returning a three.

4.43 STR 13943

A large FORTRAN task linked with DXLOBJ will no longer get an invalid SVC error >0001 when opening a named file.

4.44 STR 13990

The compiler no longer assumes the value of the variable specified by the IOSTAT= specifier is unchanged after execution of the I/O statement. The compiler used to reuse a register previously defined with the IOSTAT variable, not recognizing the definition of the variable by the I/O statement.

4.45 STR 14218

The M option has been added to the compiler to suppress printing the map information in the source listing.

4.46 STR 14220

If the first reference of an I/O unit number is in a REWIND statement, the rewind is now done.

4.47 STR 14238

The DEFINE FILE statement has been restored to FORTRAN-78 tc aid users wishing to migrate from previous implementations of TI FORTRAN.

4.48 STR 14256

The ISA subroutines RDRW and WRTRW will check that the file is open for unformatted direct access I/O and issue an error message if it is not.

4.49 STR 14292

If a program calls the ISA subroutine CLOSEW and specifies an incorrect unit number, the message output by the run time is no longer garbled.

4.50 STR 14370

When a CLOSE statement is executed in the LUNO I/O environment, the run time will release the LUNO correctly and not produce an SVC error.

4.51 STR 14371

The ISA subroutine has been fixed so that it does not result in an SVC error >0092 when used in the LUNO I/O environment.

4.52 STR 14410

A new run time library, F66OBJ, has been provided. This library contains alternate versions of some of the FORTRAN-78 library routines. These alternate versions have the same calling conventions as the equivalent subroutines implemented in past versions of TI FORTRAN, and aid users migrating to FORTRAN-78.

4.53 STR 14446

The compiler will print a variable number of lines on a page of the source listing. The number of lines per page is now a prompt on the XF78 and XF78F commands.

4.54 STR 14524

A bug in the run time module F$ROPN was fixed so that the module may be used in a write-protected memory segment.

4.55 STR 14526

FORTRAN-78 now supports default formatted sequential I/O units designated by an asterisk (*) unit specifier (e.g. WRITE(*,100)).

4.56 STR 14536

The compiler now has an optional error listing file. The compiler writes error messages, source lines that cause diagnostic errors, and module level summaries of the number of errors and warnings for each module compiled to the file or device specified by the ERROR ACCESS NAME prompt of the XF78 and XF78F commands.

4.57 STR 14540

When a FORTRAN-78 program terminates normally, it no longer displays the message "0040 NORMAL PROGRAM COMPLETION". If a termination message is wanted, the program may specify one on the STOP statement. The run time can now optionally write error and warning messages to a file or device specified by the user. none is specified, error and warning messages are written to t Terminal Local File as before. The file or device to use is specified by the ERROR ACCESS NAME prompt on the XFT and XFTF commands. There is no provision for an error file or device in the LUNO I/O environment. Error and warning messages in the LUNO IO environment are still written to the system log file.

4.58 STR 14542

When the D compiler option is used to obtain debug trace information, the compiler no longer generates incorrect code for certain cases of the RETURN statement. This error would result in an infinite loop at run time.

4.59 STR 14574

The compiler no longer generates incorrect code when generating the argument list for an OPEN statement. Sometimes the compiler used to resolve a forward jump within the argument list causing the OPEN to fail.

4.60 STR 14584

An optimization problem involving character arrays equivalenced with integer scalars has been fixed.

4.61 STR 14588

Programs may now use the OPEN statement to open a VDT with two different unit numbers (e.g. OPEN(5,FILE='ME') OPEN(6,FILE='ME').

4.62 STR 14662

The run time will release the LUNO it assigns to a file if the subsequent open SVC fails. If this is not done, and an OPEN statement with an IOSTAT= specifier is executed, and a >003B error (unable to grant requested access privileges) results, the program may recognize this and retry the OPEN after delaying a while. The final result was an SVC error >0099 - no more LUNOs available.

4.63 STR 14940

Compiler fatal error termination messages are now included in the listing output and error file listing as well as being sent to the terminal. This lets the user know what happened in case the terminal message is missed.

4.64 STR 14974

The OPEN statement will now open a magnetic tape device correctly.

4.65 STR 14983

The slash edit descriptor (/) now works correctly with ACCEPT statements where the POSITION specified is 1.

4.66 STR 15185

A problem causing the compiler to actually compute I = I - J for the statement I = -I - J has been fixed.

4.67 STR 15775

Subroutine EXTASK now returns the run ID of the task it bids as documented in the manual.

4.68 STR 16696

When a task using the LUNO I/O environment creates a logical record that is greater than the run time I/O buffer size, the run time creates the warning message correctly and no longer destroys flags needed by the read and write routines.

4.69 STR 16697

When a READ statement is executed with a format containing BN edit descriptor, the READ is correctly done and the run time does not modify arbitrary memory addresses.

4.70 STR 16698

The ACCEPT statement now terminates a field when it is full, but provides a specifier to require a field termination character to be entered. The ACCEPT and DISPLAY statements have been enhanced to allow specification of cursor blinking, alarm sounding and intensity control.

4.71 STR 16699

If the only use of a label is in an assign statement, the compiler will no longer identify the label as unused in the statement label summary.

4.72 STR 16700

If the intrinsic functions for bit manipulation are declared in an INTRINSIC statement and then passed to a subprogram as actual arguments, a unresolved reference no longer occurrs when the program is link edited.

4.73 STR 16701

If the same subprogram name appears with a different number of arguments, a warning diagnostic is issued instead of an error. This permits subroutines not written in FORTRAN to have optional arguments, provided that they do not use the run time routine F$RGMY to process arguments at run time.