VIII.4.2 SUBROUTINES NEEDED FOR A FORECAST COMPONENT OPERATION

Introduction

This Section describes the subroutines that are needed when an Operation is added to the Forecast Component.

Subroutines Needed

The following subroutines are needed for all Operations:

1. a parameter input subroutine

2. a print parameter subroutine

3. an Operations Table entry subroutine

4. an execution subroutine

If the Operation can be used for operational applications the following subroutine is also needed:

5. a card punch subroutine

Two additional subroutines are needed for Operations which have carryover values:

6. a print carryover subroutine

7. a carryover transfer subroutine

The coding for the Operation is in separate subroutines because there is only one set of subroutines which are used in all NWSRFS programs. The subroutines that are used in one program may not be used in another. Also different commands within a single program use different subroutines (e.g. the program FCINIT command SEGDEF uses parameter input and parameter print subroutines but the command PRINTSEGS uses the print parameter subroutine).

Subroutine Names

The following subroutine naming convention is used for Operations:

Subroutine Type	Name
Parameter input	PINn
Print parameter	PRPn
Execution	EXn
Card punch	PUCn
Print carryover	PRCn
Carryover transfer	COXn
Operations Table entry	TABn

where n is the Operation number

Each Operation will be assigned a unique number by the System Coordinator.

If the Operation call other subroutines unique to the Operation, they can be given any name as long as the name ends with the Operation number.

If the Operation uses common blocks to pass information from one of the primary subroutines to other subroutines associated with the Operation, these common blocks can be given any name as long as the name ends with the Operation number.

General Forecast Component Instructions for Coding Operations

There are certain general coding instructions that relate to all of the subroutines associated with a given Operation. These are listed and described in this section. Instructions related to specific subroutines are given in subsequent sections. Most utility subroutines referred to are described in detail in Section IX.3.3B. Most common blocks are described in IX.3.3C. Specific references are given for those that are not in these sections.

1. Trace Level and Debug Output Control: The Forecast Component contains a facility to trace the order that subroutines are used and to specify when debug output is to be printed. A utility subroutine is provided to check the trace level and indicate whether debug output is needed. A call to this subroutine, FPRBUG, should always be the first executable statement in each of the primary subroutines for an Operation. The form of the call is:

```
CALL FPRBUG (sname,1,n,ibug)
```

```
where sname  is the 8-character name identifying the calling
             subroutine (e.g., PIN1)
      n      is the Operation number
      ibug   is the debug output indicator:
                0 = no debug output
                1 = print debug output
```

   The trace level for all Operations is one. Trace levels equal to or greater than one specified through the program commands will cause trace messages to be printed.

   The trace message output by subroutine FPRBUG is of the form:

```
ENTER sname
```

   The person coding the Operation may also want to print a trace message when entering other Operation subroutines or exiting any Operation subroutine. If so, the trace check needed and the form of the message should be as follows:

```
      IF (ITRACE.GE.1) WRITE (IODBUG,990) SNAME
  990 FORMAT (' EXIT ‹,A)
```

The variables ITRACE and IODBUG are in the FDBUG common block. The listing of this common block is:

```
COMMON /FDBUG/ IODBUG,ITRACE,IDBALL,NDEBUG,IDEBUG(20)
```

All debug output should be printed on the unit defined by IODBUG. It is left up to the person coding the Operation to decide what debug output is to be provided. All debug output should be clear enough so that people other than the person who wrote the Operation can figure out what values are printed.

2. Error and Warning Messages: In addition to tracing and debugging facilities, the Forecast Component contains a procedure for keeping track of errors and warnings. With respect to Operations, the definitions of error and warning are:

   - An error is something that is definitely wrong (i.e., the Operation will not execute properly until the error is corrected).

   - A warning is something that may cause the answer to be wrong, but the Operation will still execute.

   When an error or warning is found, an information message should be printed. The message is written on unit IPR, indented 10 spaces and starts with **ERROR** or **WARNING**. A CALL ERROR or CALL WARN statement must follow immediately after the WRITE statement. An example is:

```
      WRITE (IPR,10) IDTQO,IDT
   10 FORMAT (‹0**ERROR** OUTFLOW DATA TIME INTERVAL (‹,I3,
     1 ‹ IS LESS THAN INFLOW DATA TIME INTERVAL (‹,I3,').')
      CALL ERROR
```

   Subroutines ERROR and WARN keep track of the number of errors and warnings and print messages indicating where the error or warning occurred (See section IX.3.0B). A Segment definition will not be written to any files if subroutine ERROR is called any time during the Segment definition.

   A STOP statement should never be used within a subroutine associated with an Operation. If calculations can not continue, print an error message, call ERROR, do any other appropriate action and return. Also, a call to Forecast Component subroutine KILL should never be used in any Operation.

3. Unit Numbers for Input/Output Devices: The unit numbers of the standard input, print and punch device respectively, are stored in the IONUM common block. The listing of the common block is:

```
COMMON /IONUM/ IN,IPR,IPU
```

   A complete description of the common block is contained in Section IX.3.0C.

Do not use either a constant unit number in READ/WRITE statements (e.g., READ(5,910)) or the nonstandard FORTRAN statements PRINT, PUNCH or READ without a unit number. Always use the variables in IONUM to specify the unit number.

4. Use of Arrays: The arrays passed through the argument lists of the subroutines associated with each Operation must be dimensioned in a very general form. Most of these arrays will be dimensioned within the subroutines as single subscripted arrays with length equal to one. For example:

```
SUBROUTINE EXn (PO,CO,PX,RO)
     .
     .
DIMENSION PO(1),CO(1),PX(1),RO(1)
     .
     .
```

   This allows the subroutines to be used in different programs without ever changing dimensions. The subroutine either determines from input variables, from parameter values previously stored or from other variables how much space actually is to be used for the given application.

   It should be remembered that the dimension statement in a subprogram merely indicates that the variable is an array and that individual values in the array will be indexed relative to the first position in the array. The location of the first position in the array is what is passed through the argument list, not the total amount of space assigned to the array. The allocation of space to an array that is passed in an argument list is made in the calling program and not in the subprogram.

   This method of using arrays is essential to the design of the Forecast Component. In the Forecast Component several arrays are used to store the information (parameters, carryover, time series data, etc.) needed by all of the Operations used for a Segment. The arrays passed through the argument lists of the subroutines for a given Operation consist of only a portion of the arrays. The length of the portion of the arrays used by each Operation can vary, depending on the application, the options being used and the data time interval of the time series.

   A complete description of the arrays used by the Forecast Component are described in Section VIII.2.

5. Work Space: In addition to space to store parameters, carryover and time series data, some subroutines for certain Operations need a certain amount of temporary work space for storing intermediate values. When the length of the work space needed varies from one application to another, the work space should be passed through the argument list just like the other arrays. In this case, the work space is dimensioned with length equal to one similar to the other arrays.

If the length of the work space needed by a subroutine is constant, the work space does not have to be passed through the argument list. However, if the amount of space needed is relatively large, it is probably a good idea to include the work array in the argument list. This allows the space to be shared by all Operations in a Segment.

Work space is most frequently needed by the execution subroutine for an Operation but, in some cases, other subroutines like the input or carryover transfer subroutines need work space.

More than one work array can be passed through the argument list if needed. No guarantees can be given on the initial contents of work space.

6. Time Series Information: Within the Forecast Component all time series have an internal identification. The internal identifier information is used exclusively to identify input and output time series for each Operation. The internal identifier information for each time series associated with the Operations for a given Segment will be unique. The internal identifier information consists of:

   - an 8-character identifier that is selected by the user
   - a 4-character data type code
   - a data time interval in hours

   Data time intervals of 1, 2, 3, 4, 6, 8, 12 and 24 hours are allowed within the Forecast Component.

   Each data type that can be used within the Forecast Component is assigned standard internal units. All standard internal units are metric. All data read from external files will be converted to these units before being used in the Forecast Component. All Operations should be written to accept time series only in these standard units. Also, all output time series generated by an Operation must be in these standard units. Operations can display information in other than the standard units.

   A complete description of the internal identification of time series, including a list of allowable data type codes and standard units, is in Section V.2.

7. Missing Data: Missing time series data are specified in the Forecast Component as -999.0 (missing time distribution is specified as -998.0). Within the Forecast Component certain time series data types are guaranteed not to contain any missing values. For data types that can contain missing values, the routine IFMSNG can be used to determine if a given time series value is equal to either of the missing data values.

8. <u>Conversion Factors</u>: The standard internal units for all Forecast Component data are metric. Operations that display information need to be able to display the values in either metric or English units. The METRIC variable in the FENGMT common block (see section IX.3.OC) is used to specify the output units. Standard conversion factors for converting from standard metric to standard English units can be obtained by using the FCONVT subroutine. In most cases it is better to obtain the conversion factors by calling FCONVT in the input subroutine and storing the conversion factors in the parameter array than to call FCONVT every time a conversion factor is needed during execution.