VIII.4.2-EX FORECAST COMPONENT OPERATION EXECUTION SUBROUTINE (EXn)

Function: The execution subroutine executes the Operation for a given time period and if needed saves carryover values.

Arguments: The argument list for this subroutine is:

```
   SUBROUTINE EXn (PO,CO,D1,...,Dn,W1,...,Wn)
```

The contents of the argument list are:

1. PO - real array dimensioned PO(*) that contains the parameters and other information needed to execute the Operation (input)

2. CO - real array dimensioned CO(*) that contain the carryover values (input and, if requested, also output)

3. D1,...,Dn - real arrays dimensioned D1(*),...,Dn(*) that contain the time series data for the Operation (input and/or output). Some Operations can access many different types of time series data (e.g. a general plot Operation) so that instead of passing individual time series in the argument list, the entire time series data array (D) is passed plus an integer array containing pointers to the first data value in each of the time series used for this application.

4. W1,...,Wn - real arrays dimensioned W1(*),...,Wn(*) that are to be used by the Operation for temporary working space (no guarantees on the initial contents)

Operations that do not have carryover will not have the CO argument. Many Operations will not need any working space. Some Operations may need some other arguments, but this will not be the typical case.

Timing Information: Timing information for the Forecast Component is contained in common block FCTIME. The contents the common block is:

```
   COMMON /FCTIME/ IDARUN,IHRRUN,LDARUN,LHRRUN,LDACPD,LHRCPD,NOW(5),
                   LOCAL,NOUTZ,NOUTDS,NLSTZ,IDA,IHR,LDA,LHR,IDADAT
```

A description of the variables and the time zone codes available in the Forecast Component is in Section IX.3.OC.

The execution subroutine performs computations or generates output from internal clock time IDA, IHR to LDA, LHR. Both the beginning and ending times represent the time at the end of a computational time interval. For example, if the total execution period starts at day 3 hour 12, ends at day 9 hour 12 and the computational time interval for the Operation is 3 hours then IDA=3, IHR=15, LDA=9 and LHR=12. The length of the execution period will be a multiple of the basic computational time interval for the Operation. Also the values of IHR and LHR will also be multiples of the computational time interval for the Operation. Some Operations need to use the end of the computational period (last time interval of observed data) in generating displays or making adjustments. The last hour of the computational period (LHRCPD) is not guaranteed to be a multiple of the computational time interval for the Operation. If LHRCPD falls within a computational period the Operation will need to determine if that period is treated as an observed data period or a future period. Generally if part of the time period is in the future the entire period is treated as being in the future.

Execution is always done on an internal clock time basis. However, the execution subroutine may need to use other clocks for:

- o output displays

- o making diurnal or seasonal adjustments

The subroutine MDYH1 can be used to convert from internal clock time to the month, day, year and hour of a specified time zone (see IX.3.OB-MDYH1).

Locating Time Series Data: The first data value in the time series data arrays [D1(1),......,Dn(1)] is for internal clock day IDADAT (see common block FCTIME) and internal hour equal to the time interval of the time series. The following equations can be used to locate a time series value for a specific day and hour:

1. The general form of the equation for locating time series data is:

   I = (KDA-IDADAT) * (24/IDT) * N + (KHR-1/IDT) * N + J

where		
	I	is the location of value in the time series data array
	KDA	is the day associated with data value being located (internal clock)
	KHR	is the hour associated with data value being located (internal clock)
	IDT	is the data time interval of the time series data
	N	is the number of values in the time series per time interval
	J	is which of the N values per time interval is being located

2. If the number of values per time interval (J) is equal to 1 the equation reduces to:

   I = (KDA-IDADAT) * (24/IDT) + (KHR-1/IDT) + 1

3. If KHR is a multiple of IDT the equation reduces further to:

   I = (KDA-IDADAT) * (24/IDT) + (KHR/IDT)

These equations should be used both to find the location of data values in the input time series for an Operation and to determine the location of where values should be placed in all output time series generated by the Operation.

Saving Carryover: Control information for saving carryover (state variables) is contained in common block FCARY. The contents the common block is:

```
   COMMON /FCARY/ IFILLC,NCSTOR,ICDAY(20),ICHOUR(20)
```

A description of common block FCARY is in Section IX.3.3C.

When determining if carryover should be saved, the first variable that needs to be checked is IFILLC. If IFILLC=0 then no carryover values are to be saved by the execution subroutine. This situation would occur if the Operation were included in a trial-and-error iterative loop. If IFILLC=1 then the execution subroutine should replace the initial carryover values in the CO array with the carryover values for the ending time, (i.e., LDA,LHR) before leaving the subroutine.

If IFILLC=1 then in addition to updating the CO array at the end of the execution period, carryover values may need to be saved at various times during the execution period. The number of times for which carryover is to be saved is defined by the variable NCSTOR and the actual times in internal day and hour are given in ICDAY( ) and ICHOUR( ). For each of these times the carryover values are saved by calling subroutine FCWTCO. This call is of the form:

```
   CALL FCWTCO (KDA,KHR,CTEMP,NUM)

   where KDA   is the current day (internal clock)
         KHR   is the current hour [KDA,KHR correspond to a day and
                 hour contained in ICDAY( ), ICHOUR( )]
         CTEMP is the carryover array containing the array CO
                 variables for KDA,KHR
         NUM   is the number of values in array CTEMP
```

A description of subroutine FCWTCO is in Section IX.3.3B.

Remember that even if NCSTOR is greater than zero no carryover should be saved unless IFILLC=1.

Carryover is saved by the Operational Forecast Program by calling subroutine FCWTCO. The updating of the CO array is only used by the other NWSRFS programs that use the Forecast Component (MCP3 and OPT3). These programs have the ability to simulate long periods by executing the Operations table for one or more Segments on a month-by-month basis. The updating of the CO array is used in these programs to transfer carryover from one month to another. When the ending hour (LDA, LHR) is one of the carryover save dates in ICDAY( ) and ICHOUR( ) both FCWTCO is called and the CO array updated for the ending hour.

Special Common Blocks: Several special common blocks are used in the Forecast Component to implement various run time options. Each of these common blocks only needs to be included in the execution subroutine for certain Operations. A list of these special execution common blocks and the types of Operations that they should be included in are as follows:

1. COMMON /FNOPR/ NOPROT - must be in all execution subroutines that can generate printer output other than error messages and debug output

2. COMMON /FENGMT/ METRIC - must be in the execution subroutine for all Operations that can generate printer output containing values that are unit (English or metric) dependent

3. COMMON /FPLTAB/ IPLHY,IPRHY - should be in all Operations that can print hydrographs in either plotted or tabular form

4. COMMON /FSNW/ NOSNOW,IPRSNW - must be in all snow model Operations

5. COMMON /FSACPR/ IPRSAC,NOFRZE - should be in all soil moisture accounting Operations that have the capability to display the accounting variables for each time interval and must be in all Operations that have the option to account for the effect of frozen ground on runoff

6. COMMON /FSNWUP/ IUPWE,IUPSC - should be in all snow model Operations that optionally can use water-equivalent or areal snow cover values to update the model computed values

7. COMMON /FPROG /MAINUM,VERS,VDATE(2),PNAME(5),NDD - should be in all Operations that do different computations for different NWSRFS programs (e.g. program logic is slightly different for the operational program than for a calibration program)

A description of these common blocks is in Section IX.3.3C (FENGMT is in Section IX.3.OC).

In addition to these execution common blocks that indicate run time options common block WHERE (Section IX.3.OC) is needed by Operations that print the Segment name as part of a display.

Run-time Modifications (MODs): The Forecast Component Execution Function (FCEXEC) of the Operational Forecast Program (FCST) contains a MOD feature which allows the user to change selected time series, carryover or other data values at run-time. Most of the MODs are implemented outside of the execution subroutines for the Operations, but in some cases the changes can only be made within the execution subroutines. In these cases special common blocks are used to specify the time and magnitude of the modifications. A list of the currently available MOD related common blocks and the type of Operations they should be included in are as follows:

1. FCOSAC - used to make modifications to the Sacramento soil moisture accounting procedure or Operations using a frost (frozen ground) index

2. FSDATA - used by snow model Operations to modify water-equivalent, areal snow cover or reliance values

3. FPXTYP - used to modify the form of precipitation (rain vs. snow) typically in a snow model Operation

4. MOD129 - used to modify API values in a API rainfall-runoff Operation

5. MOD126 - used to enter mean flow values to override reservoir model computations

Additional MOD related common blocks will be established as needed to pass run-time modifications to Operation execution subroutines.

<u>Rating Curves and Stage-Discharge Conversions</u>: For Operations that use Rating Curves subroutine FSTGQ should be used to make stage-discharge conversions or to obtain information about the Rating Curve for display purposes. Subroutine FSTGQ is described in Section IX.3.3B. This subroutine will convert stage to discharge or discharge to stage. FSTGQ can also be used to get information about the Rating Curve such as upper and lower limits or the method used in making extensions. This information might be needed by a plotting Operation so that different symbols could be used when the Rating Curve is exceeded.

General flow-point information is also stored in the Rating Curve file. If an Operation needs some of this information for computation and display purposes the Rating Curve common block (FRATNG) needs to be included in the execution subroutine. The FRATNG common block is described in Section IX.3.3C. Common block FRATNG is filled before to the call to the execution subroutine for the Operation.

<u>Special Computational Subroutines</u>: There are some computational algorithms that can be used by more than one Operation. The current subroutines that contain such algorithms are as follows:

1. FAJMDQ - adjusts instantaneous discharge values so that the resulting volume is within a specified tolerance of an observed volume

2. FSIGFG - rounds a real number to any specified number of significant figures

3. FBLEND – adjusts computed values by blending from the last observed value

A description of these subroutines is in Section IX.3.3B.

<u>Checks</u>: Most of the checks for the Operation should be included in the parameter input subroutine. For Operations with carryover a check of the initial carryover values should be made in the execution subroutine. Even though carryover values are checked in the input

and carryover transfer subroutines inconsistent values could still exist when the Operation is executed because of changes made to individual parameter and carryover values.

Comments: Some general comments related to the execution subroutine are given below.

1. The execution subroutine should be efficient both in terms of core storage and execution time. The other subroutines associated with an Operation do not have to emphasize efficiency as much as the execution subroutine because they are not used under real-time conditions.

2. The units of all values included on any printer output should be clearly identified.

3. The execution subroutine must be able to execute all versions of the Operation.