V.3.3-RES-SNGL-RCL SINGLE RESERVOIR REGULATION OPERATION RESERVOIR COMMAND LANGUAGE (RCL) SECTION

The RCL section is used to specify the controlling statements for the RES-SNGL Operation.

Reservoir Command Language (RCL) allows for different operating plans for each reservoir. Operating plans are instructions linked and triggered by logical checks on hydrologic and other conditions. The instructions are carried out by the computational routines; the links are provided by the RCL in the supervisory routine.

The maximum number of variables in the RCL is 200 (including parentheses).

The RCL is comprised of a section header, three statement types and a section trailer:

1. RCL — Section header - opens input of RCL statements
2. SET — RCL statement for assigning values to user selected variable names
3. IF-ENDIF — RCL statement block for making logical checks on existing conditions
4. DO — RCL statement for executing Schemes and Utilities
5. ENDRCL — Section trailer - closes input of RCL

Syntax description and other information for RCL section input is in subsections 1 through 5.

Examples of RCL input are in subsection 6.

1. RCL Header

The RCL header opens input of the Reservoir Command Language. The format is:

```
RCL
```

2. SET Statement

The SET statement is used to assign a constant numerical value to a user-chosen variable name. The names can then be used in relational expressions within IF statement blocks. The use of the SET statement is optional.

The format for a SET statement is:

```
SET 'a' = 'n'
```

where 'a' is user assigned name
'n' is numerical value (either real or integer) and spaces are allowed before and after the '='

The rules of the SET statement are:

1. All SET statements must precede any other type of RCL statement,

2. The variable name:

   a. Can not start with a digit (i.e., 0-9),

   b. Can not be more than 12 characters long,

   c. Can not be all blanks,

   d. Can not be one of the system comparison variable names

- QO	- SURCHARGE
- QOM	- FORECAST
- QI	- GOFLASH
- QIM	- NFLOOD
- POOL	- NSURCHARGE
- STORAGE	- NGOFLASH
- DAY	- RISING
- FLOOD	- FALLING
- OBSERVED	- RULE
	- MAXQ

3. SET variable names can only be used with system comparison variables in relational expressions having the same dimensions.

   For example, if a SET variable is first used in comparison with a system comparison variable of dimension L (for length), it must always be used with system comparison variables of dimension L. The system comparison variables that can be used with SET variables and their dimensions are:

Identifier	Dimension	Description
QO	L3/T	Instantaneous discharge
QI	L3/T	Instantaneous inflow
QOM	L3	Mean discharge
QIM	L3	Mean inflow
POOL	L	Pool elevation
STORAGE	L3 (STORAGE only)	Storage contents
DAY	DLES	Day of year (relative Julian date)

3. IF-ENDIF Statement Block

Logical checks on existing hydrologic or timing conditions are achieved with the IF-ENDIF statement block. Maximum number of IF statements and ELSEIF statements is 50. Maximum number of stack

level of embedded IF-ENDIF block is 10.

3.1. Basic Block Structure

The basic form of an IF-ENDIF block is:

```
IF (<expression>) THEN <statement_group>
ENDIF

where  <expression>      is a logical expression that has a result
                           of either 'true' or 'false'
       <statement_group> is a DO statement followed by one or more
                           DO statements and/or IF-ENDIF blocks
```

An evaluation of 'true' for the <expression> results in the statements between the THEN and END IF to be executed. A 'false' result ignores those statements and executes the statement following the ENDIF statement.

The syntax for the basic IF-ENDIF block is:

1. the IF statement must start on a new line
2. the THEN must be on the same line as the end of the <expression> (see section on logical expressions for further explanation)
3. the first DO statement must follow the THEN word (but not necessarily on the same line; no continuation needed if on next line)
4. any other statements in the <statement_group> must follow correct syntax for that statement and start on a new line
5. The ENDIF must start on a new line
6. every IF requires a matching ENDIF

3.2. Logical Expressions

The checks that can be made in an IF-ENDIF block use two types of comparisons (maximum number of comparison variables is 50):

1. Relational expressions
2. Logical variables

A relational expression compares two quantities, whereas a logical variable by itself has a 'true' or 'false' value.

3.2.1. Relational Expressions

A relational expression takes the form

<system_variable> <relational_operator> <user_variable>

The <system_variable> is one of seven allowed standard comparators:

1. QO - instantaneous discharge
2. QOM - mean discharge
3. QI - instantaneous inflow
4. QIM - mean inflow
5. POOL - pool elevation
6. STORAGE - storage contents
7. DAY - relative Julian date

When any of these comparators are used they must be spelled as listed and must appear on the left-hand side of the expression.

The <relational_operator> is one of:

1. .EQ. - left and right sides are equal
2. .NE. - left and right sides are not equal
3. .LT. - left side is less than the right side
4. .GT. - left side is greater than the right side
5. .LE. - left side is less than or equal to the right side
6. .GE. - left side is greater than or equal to the right side

The <user_variable> can be one of three types:

1. set variable name defined prior to the IF-ENDIF block in a SET statement
2. numerical value
3. system function name

SET variables are described in Section 2. Numerical values can be either integer or real.

There are two system functions:

1. RULE - current rule curve elevation
2. MAXQ - maximum discharge that can be released at current elevation. This can be referenced at different levels of definition (see requirements below)

These functions can be used as their actual values or modified by adding, subtracting, multiplying by, or dividing by a factor. For example, if the pool elevation is to be checked against the rule curve elevation plus an additional amount, the relational expression would look like:

(POOL.LE.RULE+2.0)

Requirements on the use of the system functions are:

1. If RULE is used, a rule curve must have been defined in at least one Scheme/Utility in the SPECIFIC section,

2. The MAXQ function needs to be defined at the level of definition referenced in the relational expression. For example, if this relational expression was entered,

   (QO.GT.MAXQ(2))

   the MAXQ function must have been defined at level 2 in the SPECIFIC input section.

Examples of relational expressions are:

1. Comparing the instantaneous inflow against exceeding the constant value 50000,

   (QO.GT.50000.0)

2. Comparing the mean discharge for being less than a SET variable named MAXMEAN,

   (QOM.LT.MAXMEAN)

3. Comparing the relative Julian day to the first of February,

   (DAY.GT.31)

The only syntax rule for the relational expression is that no spaces are allowed between the outermost left and right parentheses. The preceding examples all show correct syntax; the following is an example of incorrect syntax:

(QO .GT. 10000.)

Additional syntax rules on logical expressions are given in Sections 3.2.2 and 3.2.3.

Whenever a comparison is made, the system variable quantity is taken as the last value computed for that variable. So, if an IF-ENDIF block is entered before any Scheme is executed, the value used is the model result from the previous period. Since the IF-ENDIF block uses the most recent model results for testing, it will pick up the last Scheme's output in the case of stacked Scheme executions preceding an IF-ENDIF block.

3.2.2. Logical Variables

A logical variable represents the state [either 'on' ('true') or 'off' ('false')] of various hydrologic and timing conditions. The variable by itself constitutes a valid logical expression. The variables available and their 'on' conditions are:

Variable Name	'On' Condition
OBSERVED	Run is preceding or on last period of observed data.
FORECAST	Run is past last period of observed data.
FLOOD	Utility SUMINF has been executed and 'flood' criteria have been exceeded (is 'false' if no SUMINF has been executed prior to test).
SURCHARGE	Utility ENTERISC has been executed and indicates that induced surcharge is needed ('false' if ENTERISC not executed).
GOFLASH	Utility GOFLASH has been executed and indicates that flashboard Scheme is needed ('false' if GOFLASH not executed).
RISING	Pool elevation is rising from previous period to this period at a rate greater than +0.001 M/HR. If no Scheme is used, the pool elevations from previous period and previous-previous period will be used.
FALLING	Pool elevation is falling from previous period to this period at a rate less than -0.001 M/HR. If no Scheme is used, the pool elevations from previous period and previous-previous period will be used.
NFLOOD	Opposite of FLOOD.
NSURCHARGE	Opposite of SURCHARGE.
NGOFLASH	Opposite of GOFLASH.

The syntax rules and conditions for comparison are the same as for logical expressions.

3.2.3 Expression Combinations

Multiple relational expressions and/or logical variables can be combined in one logical expression by the logical operations .AND. and .OR. .

For example,

```
(QO.GT.MAXQ.AND.RISING)
```

is a valid expression.

Sub-expressions (less than the entire logical expression) can be isolated by parentheses to ensure proper order of evaluation. For example,

```
((QO.GT.MAXQ.OR.QI.GT.10000.).AND.FALLING)
```

In this example, the combination surrounding the .OR. will be evaluated and that result used with the .AND. operator and the FALLING variable to get the final expression result.

The hierarchy of evaluation (in the order of first done to last done) is:

1. expression in parentheses
2. .AND. separated expressions
3. .OR. separated expressions

All operators of equal hierarchy are evaluated from left to right.

There are a few new syntax rules when entering combined expressions:

1. there must be a balance of parentheses (i.e., an equal number of left and right parentheses and in a logically correct grouping)
2. for expressions that may exceed one line in length, the expression can be continued with an '&' as the last field on the first line and continuing the expression on the next line
3. the THEN word must be on the same line as the end of the expression

An example of a valid continuation in an expression is:

```
IF ((QI.LT.35000.AND.POOL.LT.635.AND.RISING).OR. &
   (QI.GT.35000.AND.POOL.LT.635.AND.FALLING)) THEN
   DO STPOOLQ
```

3.3. Statement Group

The first RCL in the <statement_group> of an IF-ENDIF block must be a DO statement. It can be on the same line of input as the THEN or can start on a new line. Optionally, additional RCL statements (either DO statement or IF-ENDIF groups) can be entered after the first DO statement). Any IF-ENDIF blocks imbedded within the first IF-ENDIF block are evaluated independently of the first block. There is no limit to the number of imbedded IF-ENDIF blocks.

An example of the basic IF-ENDIF block is:

```
DO ENTERISC
IF (SURCHARGE) THEN DO INDSRCHGE
ENDIF
```

3.4. ELSE Feature

Another feature of the IF-ENDIF block is an ELSE to which processing control is passed if the comparison of the IF turns out 'false'. Its use is optional but provides a clear way to specify alternative actions in a comparison.

The form of the IF-ENDIF block using the ELSE feature is

```
IF <logical expression> THEN
   <statement group 1>
   ELSE
      <statement group 2>
   ENDIF
```

If the <logical expression> is 'true' then statement group 1 is executed and control is passed to the statement following ENDIF. If the expression is 'false', statement group 2 is executed and control goes to the statement following the ENDIF. The syntax and rules for the statement group after an ELSE are the same as for that following a THEN. Only one ELSE can appear in an IF-ENDIF block.

An example of RCL employing the ELSE feature is

```
DO ENTERISC
   IF (SURCHARGE) THEN DO INDSRCHGE
      ELSE DO PASSFLOW
   ENDIF
```

3.5. ELSEIF Feature

In the case where a selection is needed between more than two possible paths of execution within an IF-ENDIF block, the ELSEIF feature can be used. The form of an IF-ENDIF block using the ELSEIF feature is:

```
IF <log. expr. 1> THEN
   <statement group 1>
   ELSEIF <log. expr. 2> THEN
      <statement group 2>
      ELSE <statement group 3>
   ENDIF
```

If <log. expr. 1> is 'true' then statement group 1 is executed and control goes to the statement after the ENDIF. If <log. expr. 1> is 'false' then control is passed to the ELSEIF where <log. expr. 2> is evaluated. If this proves 'true' then statement group 2 is executed and control goes to the statement after the ENDIF. If <log. expr. 2> is 'false' then statement group 3 is executed.

There is no limit to the number of ELSEIF options that can appear in an IF-ENDIF block. The ELSE is optional and, if used, must appear after the last ELSEIF. Only one ELSE can appear in the IF-ENDIF block.

The syntax and rules for the logical expression and statement group used with the ELSEIF follow the rules described in III.3.2 and III.3.3.

An example of the use of the ELSEIF feature (using only one ELSEIF and no ELSE) in an IF-ENDIF block is:

```
DO ENTERISC
DO GOFLASH
IF (SURCHARGE) THEN
   DO INDSRCHGE
   ELSEIF (GOFLASH) THEN
      DO PASSFLOW
   ENDIF
```

4. DO Statements

The DO statement is used to execute a Scheme or Utility defined in the SPECIFIC input section. The form of a DO statement is:

```
DO name
```

where name is the Scheme or Utility identifier and must be input on the same line as the DO

The identifiers are the same as those used in the SPECIFIC input section and listed in Table 1.

All Schemes need a DO statement for execution but not all Utilities do. Table 1 also lists those Utilities requiring a DO statement for execution.

If a Scheme/Utility is specified on a DO statement (and requires a DO) it must have been defined in the SPECIFIC input section. Conversely, any Scheme/Utility defined in the SPECIFIC input section (and requiring a DO statement for execution) must be specified on a DO statement within the RCL.

An example of a DO statement (executing the prescribed discharge Scheme) is:

```
DO SETQ
```

5. ENDRCL Statement

The ENDRCL statement closes out the RCL section. The format is:

```
ENDRCL
```

6. RCL Examples

In all the following examples, it is assumed that all referenced Schemes/Utilities have been defined in the SPECIFIC input section. Indentation is for clarity only and is not necessary.

1. The reservoir is to be operated by passing inflow.

```
RCL
   DO PASSFLOW
   ENDRCL
```

2. Conditions are to be checked for going into induced surcharge and if not met the prescribed discharge Scheme is to be used.

```
RCL
   DO ENTERISC
      IF (SURCHARGE) THEN DO INDSRCHGE
          ELSE DO SETQ
          ENDIF
   ENDRCL
```

3. Conditions are checked for going into induced surcharge and if not met the prescribed discharge and the downstream stage and pool control Schemes are executed and the minimum of those two outputs (for mean discharge) is taken as the model output.

```
RCL
   DO ENTERISC
      IF (SURCHARGE) THEN DO INDSRCHGE
      ELSE
          DO SETQ
          DO STPOOLQ
          DO SETMIN
      ENDIF
   ENDRCL
```

4. Inflow is normally passed unless the pool reaches a certain elevation or the instantaneous discharge exceeds the allowed maximum. If either of these conditions are met, the spillway Scheme is used.

```
RCL
   DO PASSFLOW
   IF (POOL.GT.109.OR.QO.GT.MAXQ) THEN
      DO SPILLWAY
      ENDIF
   ENDRCL
```

Table 1. Scheme/Utility Identifiers

Identifier	Need RCL Statement	Description
ADJUST	no	Adjustment of model outputs using observed values Utility
BACKFLOW	no	Adjustment of inflow values using observed mean discharge and pool elevation Utility
ENTERISC	yes	Determine need for induced surcharge Scheme Utility
FILLSPILL	yes	Fill and spill Scheme
FLASHBDS	yes	Flashboard Scheme
GOFLASH	yes	Determine need for flash boards Scheme Utility
INDSRCHGE	yes	Induced surcharge Scheme
MINQ	yes	Minimized discharge Scheme
MINH	yes	Minimized upstream stage Scheme
MAXQ	no 1/	Compute maximum discharge at given elevation Utility
PASSFLOW	yes	Pass inflow Scheme
POOLQ	yes	Elevation vs discharge Scheme
POWERGEN	yes	Power generation Scheme
RAINEVAP	no	Rainfall and evaporation on reservoir surface Utility
RULEADJ	no	Rule curve adjustment Utility
RULECURVE	yes	Rulecurve Scheme
SETDH	yes	Daily rate of change of pool elevation Scheme
SETDQ	yes	Daily rate of change of reservoir release Scheme
SETQ	yes	Prescribed discharge Scheme
SETH	yes	Prescribed elevation Scheme
SETMIN	yes	Select minimum quantity of model outputs Utility

Identifier	Need RCL Statement	Description
SETMAX	yes	Select maximum quantity of model outputs Utility
SPILLWAY	yes	Spillway Scheme
STPOOLQ	yes	Downstream state and pool elevation controlled discharge Scheme
SUMINF	no	Inflow summation Utility

Note:

1/ MAXQ is the only Scheme/Utility activated by an RCL statement other than a DO statement. It is used for comparisons in an IF statement.