THE HARTMAN COMPANY

9905 39th Drive NE

Marysville, Washington 98270

FAX: (360)658-1178

PHONE: (360)658-1168

E-mail: thartman@hartmanco.com

OPERATORS' CONTROL LANGUAGE

Created:	January 2, 1989
Revised:	September 1, 1998
Revision level:	2.2

1 General

1.1 What is the OCL

OCL is a high level language that offers enhanced control flexibility and function, while providing easy programmability and effective support features.

1.2 Definitions

1.2.1 Control Language

The set of characters, conventions and rules that is used for processing information and initiating automatic digital and analog control of mechanical and electrical systems.

1.2.2 Controller

A node on the network that has connections to hardware points (inputs and outputs). Points controlled through OCL.

1.2.3 DDC System

In this document a DDC (Direct Digital Control) System is the hardware equipment and software that permits physical point information to be gathered, control sequences to be processed, and an operator interface. A system is limited to those components that comprise or can access a single data base. Systems that employ multiple panels sharing only limited information are considered separate systems.

1.2.4 Descriptors

Assigned alphanumeric representations for each hardware point, variable, program, screen, report, etc. that are used to represent the point for all program and database needs.

1.2.5 Hardware Points

All physical points that are connected to the DDC system are called hardware points. These include inputs that sense temperatures or status, and outputs that control starters, valves or control signals to other devices.

1.2.6 Network

All levels (if multiple levels exist) of the communication trunk that connects controllers of a DDC system. This is **not** a LAN (Novell) in this document.

1.2.7 Panel

A node on the controls network with programming capacity. It may be a controller, router/concentrator, etc.

1.2.8 Point Names

Assigned alphanumeric representations for each system point or variable that are used to represent the point for all program and database needs. In this document, point **names** and **descriptors** may be used interchangeably.

1.2.9 Router/Concentrator (RC)

A node on the network that has no connections to hardware points, but provides network routing services to and from a group of one or more controllers.

1.2.10 System Points

Same as **hardware points**.

1.2.11 Variables

Points that do not exist in a physical sense, but are available for use by various elements of the control program are called variables. Setpoints and run times are simple examples of variables.

2 Point and Database Feature

2.1 General Programming

2.1.1 Programming Size

2.1.1.1 Programming size shall be sufficient to meet all requirements of the sequence of operations with at least **25% excess capacity** unless otherwise specified.

2.1.1.2 The OCL program space in each controller must be sufficient to support the total of the following:

2.1.1.3 20 statements of any kind for each possible DO or AO that relies on that panel for logic.

2.1.1.4 10 statements of any kind for each possible AI or DI that relies on that panel for logic.

2.1.1.5 A minimum of 300 statements of any kind at each panel or controller; such additional programming to be available for special purpose programs, self-diagnostics, reports, and/or energy accounting.

2.1.1.6 These minimums are based on typical statement lines, each containing 4 points, variables, or constants and 3 operators.

2.1.2 Program Configuration

2.1.2.1 Each panel must have the capacity of executing multiple programs. Numbers of possible programs should not be fixed, but allow for at least one program per possible system output point, and an additional five programs in each panel. The size allowance of each program should be up to the maximum capacity for all programs in the panel as outlined above. Each program must have the capacity to be activated/deactivated, edited, or created/deleted without affecting the operation of other programs.

2.1.2.2 In DDC system configurations where logic and mathematical control exist at several different network or controller levels, the control language shall exist at all of those levels and be available to meet the size requirements listed above. Rules for language use must remain constant throughout the network levels, though some specific features may be absent at certain levels. However, provision must be made to ensure accessibility to any hardware point or variable by point name at any level of control.

2.1.2.3 Each program setup permits, as an option, a special high level password in order to make any alterations to the program. If the special password option is accepted, any program change can be accomplished only by an operator connected with the special system programmers' password. The special password option can also be reversed by an operator signed on with that password level. Special password programs are so noted when viewed by a header that indicates the program requires special password for changes.

2.1.3 Point Networks To and From Other Panels

2.1.3.1 The controller network shall employ a robust network that provides automatic network services such that points located in any other controller anywhere on the network can be employed in calculations and logical expressions in a controller just as if they were located in that controller without special definitions or networking setup; except that output point commands are limited to programs located in the controller to which those outputs are connected.

2.1.3.2 The DDC configuration shall permit a minimum of four times the number of total configurable hardware points (sum of input and output point capacity of the controller) to be automatically networked into and out of each controller. Each RC (Router/Controller shall permit at a minimum of the total number of hardware points of the maximum number of controllers served to be networked in and out for use in its programs, and reports.

2.1.3.3 Where the networked point attributes are maintained within a buffer in each controller, a routine shall be provided that ensures that this database is updated at least daily. This routine shall purge points no longer required for networking in or out of the controller and ensure all points required for networking are updated regularly.

2.1.4 Preprogrammed Routines

2.1.4.1 Any preprogrammed routines except special function operators must be written in OCL format to permit easy adjustments for special conditions or to suit particular mechanical or electrical systems. Preprogrammed control routines such as optimum start/stop programs must be provided with code and documentation so required changes can be easily and effectively made.

2.1.5 Program Readability

2.1.5.1 The purpose of the Operator's Control Language is to provide functional and understandable programs such that operators with minimal computer or language training can read, understand, and troubleshoot complex control sequences. The language is therefore intended to be as readable as possible. For this, the following features are provided:

2.1.5.2 Complex Statements: Language permits Statements constructed of multiple variables and operators without a fixed limit. Also provided is a means of line continuation that allows the operator to set line breaks wherever desired.

2.1.5.3 Mixed Math and Logic in Statements: Language allows mixing of logic and math in statements and permits a math or logic expression in place of point name or variable.

2.1.5.4 No Restriction On Order of Operators in Statements: Language allows user to construct statements in any functional order of variables, constants and Operators.

2.1.5.5 Use of Descriptors and Automatic Attribute Selection: Language permits the operator to use descriptors or hardware address to enter points in programs. When hardware address is employed, editor will replace it with the description if one exists.

2.1.5.6 Program automatically selects desired attribute of point (For example the statement that begins IF FAN ON automatically engages the system to consider the status of that point. Starting the statement IF FAN ALARM engages the system to consider the alarm attribute of the point.

2.1.5.7 Minimum Parentheses Requirement: At least 8 levels of parentheses must be available without restrictions to force order of evaluation of math and logical statements, but parenthesis should not mandatory in ordinary math or logical statements. (note: the use of parenthesis may be mandatory for special operators).

2.1.5.8 Comments: Comments must be capable of being embedded anywhere in programs (including lines with expressions) through the use of special character(s) to start and stop comments.

2.1.5.9 Floating Point Math and Integers: Language must employ floating point math, but also permit the use of integers with automatic recognition of integers as floating point equivalence when required (5 = 5.0 in math function).

2.1.5.10 Line Identification: The use of line numbers in the program is discouraged as a detriment to readability. However, employing numbers in printouts and listings to identify specific lines for errors is encouraged. Automatic indents or

2.1.5.11 other special character to show a continued line or statements within BEGIN/END groups is also encouraged.

2.1.5.12 Program Printouts: System includes fast and easy method of printing programs and comments, single programs or in groups.

2.1.6 OCL Control Function

2.1.6.1 All control decisions and calculations are executed through OCL. The use of separate routines and enhancements such as interlock programs, special calculated points, or data base enhancements to meet the requirements of OCL function is not acceptable.

2.2 Point and database rules

2.2.1 Operator Overrides

2.2.1.1 A simple uniform means permits changing point operation of any system point or point from automatic (program control) to manual (direct operator) control. The operator override (manual) mode permits the operator to assign any value for the point within its defined limits or change digital point status. Each point under manual control includes an additional special color notation every time the value/status of the point is displayed.

2.2.1.2 Each time an operator overrides a point value or status a prompt permits entering a time after which the system will automatically restore the point to Auto. If no time is entered in the prompt, the point remains in override until it is manually restored to auto.

2.2.1.3 Variable points may be declared as "Manual" or "Auto" (Auto is default) points in the point database. Any that are designated "Manual" will not be considered overridden when in Manual and will not show up on manual override reports. Normally "Manual" points are so noted in point displays by a distinctive color notation.

2.2.1.4 Overridden points can be returned to normal individually or a wild card manual release command can be sent to a pre-identified controller or group of controllers. The command will automatically release all points within the selected controller(s) that are in override mode.

2.2.1.5 Each point database permits the option to require a special high level password in order to make any alterations to the database or utilize the point's override capability. If the special password option is accepted, any adjustment to the database or override of the value or status of the point can be accomplished only by an operator connected with the special system programmers' password. The special password option can also be reversed by an operator signed on with that password level. Special password points are so noted in point displays by a distinctive color notation of the point or background.

2.2.2 Point Descriptors

2.2.2.1 Points descriptors are any combination of alpha/numeric characters. Once a descriptor is assigned to a hardware point or variable, it is all that is needed to access that point for any database or program function. The system provides an easy means to allow the user to change a point descriptor such that the new point descriptor will automatically be applied everywhere in the system where the point is employed.

2.2.2.2 Each point and universal variable have assignable alphanumeric descriptors up to 12 characters in length. In addition each controller can be assigned an identification descriptor of up to 6 characters in length. Finally each Router/Concentrator (RC) can be assigned an alphanumeric descriptor of up to 6 characters in length. In this way the points in each controller can be identically labeled with the unique controller descriptors, or each point in each controller can be unique with no controller descriptor. A special separation character is used when points are referenced using controller and point descriptor as a separator.

2.2.2.3 In program edit screens or printouts, the local points shall not be shown with the controller descriptor, however, all points from other controllers shall be represented with their controller descriptor. A header for each program shall reference the controller descriptor.

2.2.2.4 When an operator is interactively accessing database, display, or programming information of a particular controller, a header item displays the controller descriptor continuously on the screen.

2.2.3 Point Limits

2.2.3.1 Each hardware or universal variable point database permits the input of maximum or minimum values for the point. If the point is commanded outside the limit by a program or manual override, it takes the value of the appropriate limit and identifies itself by color or other means in displays to indicate that it is being commanded beyond its limits. If the maximum and minimum entries are left blank, then no limits apply to the point value.

2.2.3.2 Each point database shows the where (which program, hardware connection(s), or manual command) the value of the point is input, calculated, or commanded. It also shows the actual value being input, calculated, or commanded before being limited by the maximum or minimum values.

2.2.4 Point Value Rounding

2.2.4.1 The point database for each analog point or universal variable shall permit the operator to determine the number of significant figures beyond the decimal to be displayed. The point is rounded to the number of specified decimal places each time the point is displayed.

2.2.4.2 The point will be displayed in all screen displays and printouts to the number of decimal places set by the database.

2.2.5 Point Value Format

2.2.5.1 Point database permits universal variables to be displayed in standard decimal format, time format (21:30:30 or 21:30) or in day of year format (2\1). Time formatted points are actually decimals (i.e. 21:30 = 21.5) and day of year formatted points are actually whole numbers (i.e. 2\1 = 32). These special formatted points can be used interchangeably with other formatted points in all calculations and displays.

2.2.6 Point Database Access

2.2.6.1 A quick access to the point database of each point shall be provided by the applying the cursor to the point name or value in programs and displays and entering a click or special key. Return from the database to the program shall similarly occur via a click or key sequence.

2.3 Point types

2.3.1 Digital Points

2.3.1.1 Digital (binary) points retain the value (1 for on and 0 for off) in addition to its status so that the point can be used directly in math functions. For digital points "1", "TRUE", and "ON" are interchangeable equivalents, and "0", "FALSE", and "OFF" are equivalents.

2.3.1.2 All digital output points can be reversed to the hardware outputs so that an OFF state causes voltage output and an On state cause no voltage output. All digital input points can similarly be revised to define the deactivated state as "ON" if desired.

2.3.1.3 A five character engineering unit table, including the capability for custom units is available to the operator to select a display for the status of digital points.

2.3.2 Analog Points

2.3.2.1 Analog input and output databases are easily scaled with Table functions (see TABLE section) to establish appropriate scaling factors such that the operator can assign one or more factors for different ranges of values. For example, non linear input and output devices may require several different factors and offsets to provide required accuracy over their entire operating range.

2.3.2.2 A five character engineering unit table, including custom selections is available to label analog values.

2.3.3 Analog-Digital Points

2.3.3.1 Analog-Digital points are those that utilize two digital outputs to operate an actuator (typically called floating point or two-motor devices). The analog-digital point operates identically to any other analog point except that the operator enters the actuation time, and the position is calculated by counting the time operated in each direction with resets accomplished at end points. An option permits the operator to determine if continuous driving of the actuator is provided when either end is reached.

2.3.4 Universal Variables

2.3.4.1 Universal Variables can be either digital or analog and are identical to hardware points except there is no physical connection associated with them. There is no fixed limit on the number of universal variables that can be defined in each controller. Universal variables shall have the same capacities for identification, display, manual control, program access, command, calculation/decisions, as hardwired points.

2.3.5 Controller Variables

2.3.5.1 Controller Variables are predefined points that offer full computing capacity, but very limited display or database capabilities. Controller variables are used primarily for intermediate calculations within programs. Controller variables are accessible by any program within the controller, but cannot network to other controllers. There is no differentiation between a digital and analog controller variable. A controller variable used as binary point retains the value 1 for true or on, and 0 for false or off. A minimum of 100 controller variables shall be available in each controller

2.3.6 System Variables

2.3.6.1 In addition to hardware points and variables, the system must automatically provide points that can be used by programs as follows:

2.3.6.2 Time Of Day: TIME A system variable that automatically provides the time in standard format (HR:MN) but carries the time as a decimal (HR.HR) for calculation. TIME is based on a 24 hr clock, and automatically adjust for the Daylight Savings Time switch.

2.3.6.3 Day Of Week: DOW The day of week system variable providing the numbers (1-7) for the days of the week, switching every night at midnight.

2.3.6.4 Month Of Year (1-12): MONTH

2.3.6.5 Day Of Month (1-31): DOM

2.3.6.6 Day of Year (1-366): DOY

2.3.6.7 Date (1\1\95 to 31\12\99): DATE Note this number is actually retained as a whole number.

2.3.6.8 HOLIDAY Set by annual schedule sets DOW to 8 or 9

2.3.6.9 Time of Morning Sunrise (HR:MN): SUNRISE

2.3.6.10 Time of Evening Sunset (both HR.HR & HR:MN): SUN

(Note: both SUNRISE and SUNSET require localization factors for correct operation).

3 Programming Language Features

3.1 Program control

3.1.1 Timing Functions

3.1.1.1 OCL provides time based program sequencing in a readable fashion. This mechanism shall operate segments of a single program at precise timed intervals. The recommended operators are:

```
DOEVERY nn M(S)(H)


ENDDO
```

where the lines between the DOEVERY and ENDDO statements are executed every nn minutes(M), seconds(S) or hours(H). the term nn can be either a constant or a variable.

example:

```
DOEVERY 10 M
        Statement 1
        Statement 2
ENDDO
```

Statements 1 and 2 would be executed on the first scan of the program, and once every 10 minutes thereafter.

3.1.2 Timers

3.1.2.1 Variables acting as timekeepers must be available for user constructed timing functions. Timers can be points that are specifically available for the purpose, or as an option for variables such that unless reset to a specific value by a manual or program command, they time in hours and hundredths of hours, or minutes and hundredths of minutes to at least 9,999 hours (minutes).

3.1.3 Branch Commands

3.1.3.1 Branch commands are not encouraged for readable programs, but they are occasionally necessary in most applications. The language must support GOTO, GOSUB/RETURN, CALL, and EXIT as follows:

3.1.3.2 GOTO causes the program execution to jump immediately to the line label indicated. GOTO statements should be restricted to jumps forward in the program.

```
GOTO (line label)
```

3.1.3.3 GOSUB causes the program execution to jump immediately to the subroutine indicated. program execution returns to the statement following the GOSUB statement as soon as a RETURN statement is encountered in the subroutine.

```
GOSUB (Subroutine name)
```

3.1.3.4 CALL causes the current program execution to halt, and the program indicated to be executed. When the last line of the called program has been executed, execution returns to the statement following the CALL statement.

```
CALL  (Program Name)
```

EXIT LOOP causes the execution of the program to move immediately to the statement following the next END or ENDO in the program. If none follow, then it has the same effect as EXIT PROGRAM.

```
EXIT LOOP
```

3.1.3.5 EXIT PROGRAM causes an immediate end to the program execution. It produces the same result as reaching the last line of the program. Program execution is halted, and only restarted when conditions for the program execution are next met.

```
EXIT PROGRAM
```

3.2 Mathematical expressions

3.2.1 General Rules

3.2.1.1 Essential to the OCL language is the evaluation and programming of complex math expressions in a straightforward manner consistent with standard mathematical syntax. The language allows expressions to be written free form in standard mathematical notation.

3.2.1.2 Full 32 bit floating point arithmetic is supported and transparent to the language. Integer conversion is transparent such that they can be mixed with Decimals in calculations whether the integer is a constant or variable.

example:

```
MASP = 99.5 - 2 * AVEST - OAT / 5
```

3.2.1.3 Mathematical operators and functions must be allowed to mix with digital variables/points. Digital variables/points will equate to 1 for true/on and 0 for false/off in any mathematical expression.

example:

```
TBSP = 70 + 5 * CHILLER
```

where CHILLER is a digital status point indicating chiller operation.

3.2.1.4 Mathematical operators and functions must be allowed to be included in conditional expressions (see also Conditional Expression section).

example:

```
IF MIN(ST1,ST2) < (NLLTEMP + LOWTSRT * 2) THEN LOWSTRT
= ON
```

3.2.1.5 Variables can utilize their present value in calculations (ie: VAR1=VAR1+1).

3.2.2 Standard Math Operators

3.2.2.1 The following math operators are supported by OCL:

1. Mathematical Operators

   a. Exponentiation (^ , **)

   b. Negation (-)

   c. Multiplication and division (* , /)

   d. Modulo arithmetic (MOD)

   e. Addition and subtraction (+ , -)

3.2.3 Special Mathematical Functions

a. Below is a list of these functions

- AVE( ) [returns average value of group of variables or numeric expressions]
- MAX( ) [returns maximum value of group of variables or numeric expressions]
- MIN( ) [returns minimum value of group of variables or numeric expressions]
- ABS( ) [returns absolute value of variable or numeric expression]
- LMT( ) [limit variable to set range between variables or numeric expressions representing minimum and maximum allowable values]
- RND( ) [rounds off value of variable or numeric expression to nearest integer value]
- TRN( ) [converts decimal to integer by dropping decimal portion of number]
- TON( ) [time point has been on(true)]
- TOF( ) [time point has been off(false)]

- SIN( ) [returns sine of the angle variable or numeric expression]
- COS( ) [return cosine of the angle variable or numeric expression]
- TAN( ) [returns tangent of the angle variable or numeric expression]
- LN( ) [returns the natural logarithm of a variable or numeric expression]
- MOD( ) [returns the remainder of an integer division]
- INT( ) [returns the largest integer less than or equal to the variable or numeric expression]
- TBL( ) [returns corresponding value from user defined lookup table with at least 10 steps]

3.2.3.1 These functions return values for use in mathematical or conditional expressions.

example:

A = C + D + MIN(E,F)

3.2.3.2 Mathematical expressions are permitted in place of variables in all mathematical functions.

example:

B = MAX(C,A+B)

3.2.3.3 Or a combination

D = MIN(E,AVE(F,G,H+I)) + J

A. Order of Precedence

3.2.1.1 Math statements are be evaluated in a fixed order of precedence unless changed by parentheses; math statements must always be evaluated before relational and logical operators in conditional statements. Statements must be evaluated in the following order:

1. Arithmetic operations
   a. Exponentiation (^ , **)
   b. Negation (-)
   c. Multiplication and division (* , /)

d. Modulo arithmetic (MOD)

e. Addition and subtraction (+ , -)

2. Relational operations

a. (= , <, > , <=, >=, <>)

3. Logical operations

a. AND, OR, NOT

3.2.1.2 Operators at equal levels in a statement are evaluated left to right.

example:

VALUE = 3 + 6 / 12 * 3 - 2 is evaluated:

a. 6 / 12 (= 0.5)

b. 0.5 * 3 (= 1.5)

c. 3 + 1.5 (= 4.5)

d. 4.5 - 2 (= 2.5) VALUE = 2.5

B. Parenthesis in arithmetic operations

3.2.1.1 Parenthesis change the order in which arithmetic operations are performed. Operations within parentheses are performed first. Inside parentheses, the order of precedence is maintained.

3.3 Conditional expressions

3.3.1 General Rules

3.3.1.1 No restrictions are placed on statements executed by conditional expressions that do not apply to statements by themselves.

3.3.1.2 No additional restrictions are placed on the use of math expressions when used in conditional statements.

3.3.1.3 Mixed math and Logic allowed in IF statements

example:

```
IF   FAN1  =  ON  AND  (OAT + 5)   >>  RAT   THEN
```

3.3.1.4 In Complex logic statements, math operators are evaluated first, then relational operators, and finally logical operators. Operators of the same level must be evaluated in straight left to right order unless changed by parentheses. See Order of Precedence section under Mathematical Expression.

3.3.2 Logical Operators

3.3.2.1 **(check subheads below- do we want them?)**

```
ON        IF   FAN = ON   THEN ...
                   IF   FAN   THEN ...
TRUE      IF   FLAG1 = TRUE   THEN ...
                   IF  FLAG1  THEN ...


OFF       IF   PUMP2 = OFF   THEN ...
FALSE     IF   FACTOR2 = FALSE   THEN ...
```

(Note: "TRUE' is equivalent and interchangeable with "ON," and "FALSE" is equivalent and interchangeable with "OFF". "TRUE" or "ON" is implied when not included.)

```
AND       IF   FAN = ON   AND   FLAG1 = TRUE   THEN
OR        IF   FAN = OFF   OR   FLAG1 = TRUE   THEN
NOT       IF    NOT   FLAG1   THEN ...
BETWEEN   IF  TIME BETWEEN   3:00,   4:00    THEN
IF   OAT   BETWEEN   50,   60  THEN  ...
```

3.3.2.2 (Note: When "BETWEEN" statements are used with the system variable TIME, they must automatically adjust for the time change at midnight so the statement

```
IF   TIME   BETWEEN 17:00,  08:00   THEN  ...
```

is true between 5:00 p.m. and 8:00 a.m. the next morning.

ONTIME>	IF FAN ONTIME > nn M(S)(H) THEN ...

3.3.2.3 This expression is true only when FAN is turned on, and has been on for more than nn minutes, where nn is either a constant or variable and the term following nn is either M, for minutes, S, for seconds, or H, for hours.

ONTIME<	IF PUMP3 ONTIME< TIME1 M THEN ...

3.3.2.4 This expression is true only when PUMP3 is on, but has been on for less than the variable TIME1 minutes.

OFFTIME<	IF FLAG1 OFFTIME< 30 S THEN ...

3.3.2.5 This expression is true only if variable FLAG1 has been off for less than 30 seconds.

OFFTIME>	IF PUMP4 OFFTIME> TIME2 H THEN ...

3.3.2.6 This expression is true only if PUMP4 is off, and has been off for more than TIME2 hours.

3.3.3 Math Operators

<	IF OAT < 55 THEN ...
>	IF SPACETEMP > 70.5 THEN ...
=	IF VARIABLE1 = 8 THEN ...

(Note: "=" can be used as both a math and logical operator. In both instances it denotes equivalence)

Conditional math operators can be paired to apply further conditions:

IF A >= B (If A is greater than or equal to B)

IF A <= B (If A is less than or equal to B)<M=>

IF A <>B (If A is not equal to B)

3.3.4 Nesting Conditional Expressions

3.3.4.1 At least 10 nesting levels of if-then must be allowed. In nested IF THEN statements, ELSE always refers to last IF THEN as follows:

Check this nesting

```
IF   Expression1
     THEN
            IF   Expression2
                   THEN   Statement3
            ELSE   Statement4
ELSE   Statement5
```

In the above expression, Only if expression1 is true will expression2 be evaluated. If expression2 is true then statement3 will be executed if expression2 is false then statement4 will be executed. If expression1 is not true, expression2 and statements 3 & 4 will be ignored and statement 5 will be executed.

3.3.5 Multiple Statements - BEGIN/END

3.3.5.1 To avoid the need for jump statements or subroutines, provisions must be made to allow multiple statements to be executed conditionally within the IF THEN ELSE format. When BEGIN and END are used following THEN or ELSE, all statements between will be executed as a single statement:

```
IF   Expression1
THEN
       BEGIN
       Statement 2
       Statement 3
       END
 ELSE
       BEGIN
```

```
Statement 4
Statement5
END
```

Statements 2 and 3 will be executed only if the Expression1 is true. Statements 4 and 5 will be executed only if Expression1 is false.

3.3.5.2 The program notes each line within an IF statement by automatic indents when lines are part of a BEGIN/END group. This indication shows in the printouts of the program, and whenever the program is displayed on the screen. A suitable syntax error signal alerts the operator if the BEGIN's and END's do not match.

3.3.6 Special Conditional Statements

```
IFONCE        IFONCE   FLAG1   =   TRUE ...
```

This statement is true only the first time it is scanned after FLAG1 changes from false to true. FLAG1 must return to false and back to true for the statement to become true again.

```
IFONCE   FAN1   =   ON
```

This statement is true only for the first scan of the expression after FAN1 turns on.

3.4 Commands

3.4.1 Digital Commands

3.4.1.1 START and STOP are used to command digital points to their activated or normal states. Commands are issued only when the program in which the command lines reside has completed execution.

START(FAN1) commands point named FAN1 to its activated state.

OCL employs the principle of math equivalence such that the following statements are equivalent:

```
(1)     START(FAN1)              STOP(VARIABLE1)
(2)      FAN1  = 1               VARIABLE1  =  0
```

(3) FAN1 = ON VARIABLE1 = OFF

Expressions on lines 2 and 3 are not recommended for use in turning hardware points on and off, but are sometimes useful in making variable expressions more understandable.

3.4.1.2 The language shall allow multiple points to be commanded in a single statement:

START(FAN1,FAN2,FAN3,...)

3.4.2 Analog Commands

3.4.2.1 Analog output points are commanded to outputs using math expressions. Actual output values must be user set by scaling factors in the point database **Fix this ref.** (see section 2.1 C).

VALVE1 = 100

This command sends the output to VALVE1 represented by 100 (100% if so set by scaling factor).

VALVE2 = 50

This command sends the output to VALVE2 represented by 50.

3.4.2.2 Analog output points shall be configured to allow scaling and inverse factors so that a system with both normally open and normally closed valves with differing operating ranges can be scaled to all read and be commanded from 0-100% Open at the operator's terminal.

3.4.3 Operator Commands

3.4.3.1 Hardware points and variables operate from program when in the "AUTO" mode and respond only to operator commands in "MANUAL" mode.

3.5 Schedules

3.5.1 General Requirements

3.5.1.1 The Schedule system shall be easy to understand and easy to operate. Equally important, the schedule system shall interface with the programming language such that time elements are readily accessible. For example, the decimal value of the start and stop times must be available to the OCL program so that anticipatory comparisons and calculations based on their values can be made.

3.5.1.2 The schedule system shall employ full screen display and editing. Each schedule requires a unique descriptor or title to identify the schedule on screen. A full week, including holiday and special day, must be displayed on one screen. Simple procedures allow a schedule to be created, edited, or deleted with ease.

3.5.2 Annual Schedules

3.5.2.1 An annual schedule calendar mechanism shall be provided to schedule Holidays and Special days a minimum of one year in advance. A full screen editor in calendar format must be provided to allow speedy selection and review of holidays and special days.

3.5.2.2 Holidays, when in effect, force all schedules from standard day of week to the holiday times of each individual schedule. The program language shall have access to this holiday status.

3.5.2.3 Special days, when in effect, force all schedules that have time elements assigned in the special day schedule to adhere to that special schedule. If a weekly schedule has no elements assigned to the Special Day, it will remain on the normal day of week schedule.

3.5.3 Weekly Schedules

3.5.3.1 The weekly schedules screen shall display a full week with holidays and special days included. The scheduling shall be done "spreadsheet" style, moving the cursor to the appropriate time cell and typing in the new time. The Occupancy and Vacancy time cells shall be directly accessible from the operators control language and thus assignable to virtual points of the system.

3.5.4 Monthly Schedules

3.5.4.1 An option is to allow the choice of monthly schedules. The monthly schedules screen shall display a full month with holidays and special days included. The scheduling shall be done spreadsheet style, and shall be displayed and edited in the same fashion as the weekly schedule. The occupancy and vacancy times must again be directly accessible from the operators control language.

3.5.5 Special Features

3.5.5.1 The schedule mechanism shall have an override feature which will override a specific time cell for a single occurrence. The cursor could be placed on top of the time element to be overridden, and the override command invoked. The override time is entered and takes precedence. After a single execution, the override is deleted automatically and control returned to the original cell.

3.5.5.2 The schedule mechanism shall have a temporary feature. A temporary time element may be added to any schedule cell. It will automatically be deleted after a single execution.

3.6 Modulating controls

3.6.1 PID Controllers

3.6.1.1 A simple means of establishing controllers that provide proportional plus integral and derivative (PID) control shall be provided. These controllers must have the following features:

3.6.1.2 All factors in PID controllers must be entered and expressed as gains, limits, etc. and in terms with physical units.

3.6.1.3 Actual control Algorithm and its operation must be documented in the operator's manual.

3.6.1.4 Capacity must be provided to have different gains at different values of the proportional error signal.

3.6.1.5 The ranges of Integral and Derivative gains must be sufficiently high so that certain kinds of loops can be "jogged" into setpoint by the controller.

3.6.1.6 Convenient integration into the control language so that the use of each controller is apparent in the language code.

3.6.1.7 Capacity for coordination of multiple controllers for a single point and/or overrides so that the language can switch controllers for an output under certain conditions or override all controllers with a specific value to the output under other conditions.

3.6.1.8 Capacity for any or all Controller factors to be system variables.

3.6.2 Expert Controllers

3.6.2.1 In addition to simple PID controllers, controllers shall be available for modulating control that are expert controllers. These controllers shall operate nonlinear as well as linear control loops with continuous automatic self-tuning characteristics. Such controllers may use the standard PID controller and incorporate additional self-tuning features, or they may be entirely apart from the PID controllers.

3.7 Global commands and instructions

3.7.1 Global Commands

3.7.1.1 The continuing trend toward highly distributed logic, calculation, and I/O control in DDC systems creates a potential for database and control problems that shall be resolved by additional features of the system and programming language. To satisfy the requirements of highly distributed systems operator commands shall accept the "*" and "?" wild card features prevalent in computer operating systems.

3.7.1.2 Commands shall allow wild card characters in the point names such that an entire class of points can be commanded with a single expression

example:

```
START(*~BOXFAN)
```

3.7.1.3 This expression would start the point in any controller that is labeled "BOXFAN". Note that the "*" wild card indicates any controller identification is with the command set and the "~" is the special character that separates the controller descriptor from the point descriptor

```
START(B1??~BOXFAN)
```

3.7.1.4 This expression would start any point labeled "BOXFAN" whose controller identifier starts with "B1" and has two more characters in its label.

```
START(B1*~BOXFAN)
```

3.7.1.5 This expression would start any point labeled "BOXFAN" whose controller identifier starts with "B1" and has one or two more characters in its label.

```
START(B1*~???FAN)
```

3.7.1.6 This expression would start any point whose label ends in "FAN" and whose controller identifier starts with "B1" and has one or two more characters in its label.

3.7.2 Loop Calculations

3.7.2.1 Loop calculations allow a large number of commands or calculations to be performed with a minimum of program space. Because of network considerations, Loop use is primarily aimed at Router/Concentrator (RC) nodes. Calculation loops incorporate the wild card feature also.

example:

```
CV1 = 0
```

```
CV2 = 0
CV3 = 0
CV4 = 100
FORALL = ROOMTEMP*
        CV1 = CV1+1
        CV2 = CV2+(ROOMTEMP*)
        CV3 = MAX(ROOMTEMP*, CV3)
        CV4 = MIN(ROOMTEMP*, CV4)
ENDFOR
AVEST = CV2/CV1
MAXST = CV3
MINST = CV4
```

This expression will calculate the average, maximum and minimum of all points within the RC network whose descriptors start with "ROOMTEMP"

3.7.2.2 The above example will not search points out of the RC network because no prefix was provided. If the statement read:

```
FORALL =*~*~ROOMTEMP*
```

then all controllers on all networks that are at or below the network of the RC in which the program resides will be searched for point match, and all points in those controllers or RC that begin with "ROOMTEMP" will be included in the calculation.

3.7.3 COPY Commands

3.7.3.1 The COPY command shall permit the transfer of complete databases, or certain specified portions such as individual programs, controllers, screens, point databases, descriptors, or other database information to any level of controller or RC. These commands are:

```
COPY file spec  controller(or RC)
```

This command causes an entire database labeled "file spec" on the system disk to be downloaded to a controller or RC labeled "controller" or "RC". Wild cards can be used to accomplish a large program upgrade. If a wild card format is used, a prompt will be returned to ask if the operator would like to confirm load to each controller.

3.7.3.2 Exclusions can be included in the load instructions such as:

```
COPY file spec   controller  EXCEPT POINTS PROG1 PROG2
```

This permits new programs, screens, reports, etc. to be downloaded to a controller or group of controllers whose point database has already been defined and calibrated and whose programs 1 and 2 may include special functions that are unique to each controller in the group. Other exclusions include:

EXCEPT SCREEN1 (or SCREENS) REPORTS (or REPORT3) SCHEDULE2

3.7.3.3 Also limiting loads is the ONLY statement for example:

```
COPY file spec  controller \ONLY PROG4 PROG5
```

will load only programs 4 and 5 from the database "file spec" into the controller or RC.

3.7.3.4 The COPY command can also be employed to transfer a database from and existing controller or RC on the network to another:

```
COPY controller1 controller2 \ONLY PROG4 PROG5
```

In this example programs 4 and 5 will be copied from controller 1 to controller 2.

3.7.3.5 The COPY command can be employed to transfer a database from and existing controller or RC on the network to a file for backup purposes:

COPY controller file spec

or it can be employed to copy portions of the database to disk for redistribution to other controllers:

COPY controller file spec \ONLY PROG4 PROG5

In this example programs 4 and 5 will be copied from controller to a file.

3.7.4 CLEAR Commands

3.7.4.1 The CLEAR command is used to remove all or portions of an existing database. The CLEAR command is implied in the COPY command, but if no new program or database item will replace an existing obsolete item, the CLEAR command will remove the existing:

CLEAR controller \ONLY PROG4 PROG5

This command clears programs 4 and 5 from the controller.

CLEAR controller

This command clears the entire database of the controller. All outputs remain in the last commanded state when this command is issued.

3.8 Program support features

3.8.1 Program Editing

3.8.1.1 The DDC system shall include a means of easily editing OCL programs, schedules, point databases, screens, etc. with a full screen editor. Programs can be edited on-line or written externally. Systems shall be capable of accepting programs from ASCII files that have been prepared on other computers or word processors and entered with the COPY command or another command for the purpose.

3.8.2 Program Debugging

3.8.2.1 A method shall be provided of troubleshooting program errors, both of syntax and run time errors that notifies the operator as to the location of the line which is at fault. It is recommended that a trace feature also be provided to allow the operator to troubleshoot program problems. The program must provide self diagnosis for syntax errors during editing, and provide run time errors that include at a minimum:

A. Endless Loops

B. Run time Execution Errors

3.8.2.2 The system shall employ means to recover from OCL run time errors without a general panel or program failure.

3.8.3 Program Disable

3.8.3.1 A method of Disabling Program sections for startup and debug shall be provided.

3.9 Types of passwords

3.9.1.1 The system permits restriction access to portions of the network, and restricting levels of access in the network via a password system.

3.9.1.2 Each controller, RC, hardware point, variable, program, screen, schedule, report, etc. may have an individually assigned password to extend or limit access for editing or modification purposes.

3.9.1.3 A minimum of seven basic password levels shall be provided as follows:

3.9.1 Occupant

3.9.1.1 This password limits access to the controller to which the connection is made. Only those points, programs, etc. specifically designated with the Occupant level password in their database within that controller may be overridden.

3.9.2 Mechanic

3.9.2.1 This password extends access to the entire portion of the system that is not under special access restriction by one of the special passwords. Each password of this level can be customized to include or exclude the capacity to perform manual overrides on a point by point type, except that those points whose database includes a higher level password cannot be overridden by a mechanic level even if the customized setup permits that override. The mechanic level does not permit editing of points, programs, screens, etc.

3.9.3 Operator

3.9.3.1 This password extends access to the entire portion of the system that is not under special access restriction by one of the special passwords. Each password of this level can be customized to include or exclude the capacity to perform manual overrides on a point by point type, editing of point database, programs, screens, and reports, except that those points, programs, etc. whose database includes a higher level password cannot be overridden by an operator level even if the customized setup permits that override. The Operator level cannot create new database items such a points, programs, or operator passwords. The operator level is the default password level for point overrides.

3.9.4 Programmer1

3.9.4.1 This password extends access to the entire portion of the system that is not under special access restriction by one of the special passwords. Each password of this level can be customized to include or exclude the capacity to perform manual overrides on a point by point type, editing of point database, programs, screens, and reports, except that those points, programs, etc. whose database includes a higher level password cannot be overridden by a Programmer1 level even if the customized setup permits that override. The Programmer1 level is the default level for creating and editing database items such a points, programs, or operator passwords.

3.9.5 Programmer2

3.9.5.1 This password extends access to the entire portion of the system that is not under special access restriction by one of the special passwords. Each password of this level can be customized to include or exclude the capacity to perform manual overrides on a point by point type, editing of point database, programs, screens, and reports, except that those points, programs, etc. whose database includes a higher level password cannot be overridden by a Programmer2 level even if the customized setup permits that override.

3.9.6 Network Manager

3.9.6.1 This password extends access to the entire system including those controllers and RCs that are under special access restriction by one of the special passwords. Each password of this level permits creation and editing of the entire database except that those points, programs, etc. whose database includes the highest level password cannot be overridden by a network manager level.

3.9.7 Manufacturer

3.9.7.1 This password extends access only to the points, programs, schedules, screens, or reports in the system including controllers and RCs that have been specified with this high level password. The purpose of this password is to ensure the integrity of special manufacturer programs that may be operating boilers, chillers or other special equipment. The Manufacturer level password can be used to restore a point program, controller, etc. to be returned to the standard password level. No database element with the manufacturer password element can be cleared or copied over by an operator signed on at some other level.

3.9.7.2 The system shall also accommodate a minimum of twelve special password groups that can be used to restrict any level of password operator from access to portion of the system network. To access such a segment of the system, the operator would sign on in the normal fashion, and then provide an additional password that would gain access to portions of the system that are part of the group of the password. In this fashion, individual floors, or buildings can be isolated from personnel operating other buildings or floors.

3.10 Display screens

3.10.1 Type of Screen

3.10.1.1 A type of system point called a "Screen" shall permit programmers or operators to place text, local, global, or real point values or attributes anywhere on a screen point. Screens shall be configurable to accommodate point data, variables, and general text in any order.

3.10.2 Number of Screens

3.10.2.1 The number and complexity of screens shall be limited only by available memory. Each screen may consist of more than a single page, which is accessed by "page up" "page down" keys.

3.10.3 Updates

3.10.3.1 The value, status, and attribute data on display screens shall be automatically updated periodically.

3.10.3.2 When it is desired to manually change a value or status of a point or variable through the screen point, the data displayed on screens can be overwritten. This feature can be enabled or disabled for each point or text unit when the screen is set up depending on the parameters chosen when it is entered. Points that can be manually altered are displayed differently than points that cannot. Points that are manually overridden also show differently.

3.10.3.3 Variables in display screens may have the source of current value or status automatically displayed (program, manual, or set by other nodes on the network).

3.11 Reports

3.11.1 “SEND REPORT”

3.11.1.1 This command shall enable information to be directed to printers or file servers connected to the system. The SEND REPORT can originate in any controller or RC and send reports from any other combination of controllers and RCs. Wild card features can be employed to send groups of reports with a single command. when sending to a file, an "APPEND" switch allows an existing file to be added to if one exists. The filename can include special characters that will automatically change the filename based on time/date or changes of some other variable.

3.11.1.2 A SEND REPORT command can be manually executed, or automatically executed from an OCL program.

3.11.2 Format

3.11.2.1 The format of each report shall be custom developed. It can include a custom text, points, programs, logs, or screens. It can also include a listing of points that are within a certain range of values or status.

3.11.3 Logs

3.11.3.1 Each controller and RC shall include logs that are available to be viewed as screens or sent as reports. The logs include:

3.11.3.2 A password log that contains the password holder's ID and the last ten sign-on/sign-off for the physical connection through that controller.

3.11.3.3 An Override log that contains the password holder's ID and the last ten modifications such as point overrides, program editing, etc. that have been applied within the controller.

3.11.3.4 An alarm log that contains the most recent ten alarms and the ID of the password employed to acknowledge each.