

**Examples:**

```
LET a=1           'this will return an error because of insufficient spacing
LET a = 1        'this is valid
LET a      =    1      'this is valid
```

Only a single variable is allowed on the left side of any operator. The following example is incorrect and will return an error. The error occurs because there is more than a single variable to the left of the comparison operator (a + 2 is to the left of =).

```
IF a + 2 = 3 THEN
```

To fix the above line, replace 'a + 2.' One of the following options may be used:

```
IF a = 1 THEN
```

or

```
LET b = a + 2
IF b = 3 THEN
```

**Supported Statements:**

The following are the statements supported by the ControlByWeb™ BASIC interpreter, a short description of the statements and their formats.

**LET** - The LET statement assigns a variable a value. The format is:

```
LET (variable) = (expression)
```

**IF THEN, ELSE, END IF** - The IF THEN statement tests the truth of a condition. The ELSE statement defines a second function if the condition is found false. In other words, if the condition is true, then a function is performed. If it is not true a second function may be performed. The second function may or may not be necessary depending on the application. The IF THEN (ELSE) statement must always be followed with an END IF statement. The format is:

```
IF (variable) (=, <, >, <=, >=, <>) (expression) THEN
    (Function 1)
ELSE
    (Function 2)
END IF
```

Note that in most BASIC interpreters 'Function 1' (see above) may be placed after the THEN statement. This interpreter requires 'Function 1' to be put on the following line.

**FOR TO, NEXT** - The FOR TO statement loops a section of code a predefined number of times. The NEXT statement always follows the section of code to be looped. The format is:

```
FOR (variable) = (expression) TO (expression)
    (code to be looped)
NEXT (variable)
```

**DO WHILE, LOOP** - The DO WHILE statement loops a section of code while a condition is found true. The LOOP statement always follows the section of code to be looped. Note that if the condition is omitted, the code will be looped without end. The format is:

```
DO WHILE (variable) (=, <, >, <=, >=, <>) (expression)
    (code to be looped)
LOOP
```

**LOG** - The LOG statement causes the device to log data according to the settings specified under the **Logging** setup tab. Note that in order to log, logging must be enabled in the **Logging** setup tab. The format is:

```
LOG
```

---

**EMAIL** - The EMAIL statement causes the device to send an email. The format is:

*EMAIL (corresponding temperature sensor variable)*

The temperature sensor assigned governs the email settings used when sending the email. For example if the code stated:

EMAIL temp3

The unit would send an email triggered by temperature sensor 3 to the email addresses selected under the Sensor 3 settings in the **Sensors** setup tab. The email contains the temperatures on all sensors, and specifies which sensor triggered the email.

**END** - The END statement ends the main body of code.

**CALL** - The CALL statement is found within the main body of code, but requires the interpreter to skip to a subroutine found at the end of the program. After the subroutine is finished, the interpreter returns to the line immediately following the CALL statement. The format is:

*CALL (name of subroutine)*

**SUB, END SUB** - The SUB statement defines the beginning and name of a subroutine. The END SUB statement defines the end of the respective subroutine. Subroutine names can be up to 20 characters long and are case sensitive. The SUB and END SUB statements always must follow the END statement. The format is:

*END*

*SUB (name of subroutine)*

*(contents of subroutine)*

*END SUB*

**REM OR ' -** The REM (or ') statement designates remarks made by the programmer. The interpreter will disregard any characters following these statements.

**PRINT** - The PRINT statement outputs a variable or message (if contained within quotation marks). Note that the PRINT statement is only available for the PC version of the BASIC interpreter. The device will not recognize the command. The format is:

*PRINT (variable)*            'output variable value to screen

*PRINT "(message)"*        'output '(message)' to screen

### Variables:

Two types of variables are available for use in the ControlByWeb™ BASIC interpreter - user defined variables and useful predefined variables.

#### User Defined Variables:

Up to 10 user defined variables are allowed to be used in scripts. Variables are represented by single, lower-case characters ranging from the letter 'a' to the letter 'j' in the English alphabet. They are always global and stored internally as floating point numbers. Variables are defined using the LET statement.

Examples:

Let b = 5                            'variable b will be set to 5

Let d = b + 2                        'variable d will be set to 7

The following are useful predefined variables for the ControlByWeb™ BASIC interpreter. Note that some models may not contain digital inputs, analog inputs and/or relays and thus would not have digital input, analog input and/or relay variables.

#### Timer Variables:

Six timers are available for use in BASIC scripts. They are designated as t0 through t5. Timers can be set to any positive integer (or 0) by using the LET statement. As soon as a value is

---

---

assigned to a timer, it will begin to count down immediately by decrementing one count every 100ms until it reaches zero.

Examples:

```
Let t3 = 1500      'set timer 3 to 150 seconds
Let t1 = 0         'disable time 1
```

#### Relay Variables:

The internal relays can be turned on, turned off, pulsed, toggled, or read in BASIC scripts. The LET statement is used to set the relay state. The state options available are 0 (turn relay off), 1 (turn relay on), 2 (pulse relay), and 5 (toggle relay). Note that the relay pulse time is specified in the **Relays** setup tab. Relays are designated as relay1, relay2, relay3, etc.

Examples:

```
Let relay1 = 1      'turn on relay 1
Let relay3 = 0      'turn off relay 3
Let relay1 = 2      'pulse relay 1
Let relay3 = 5      'toggle relay 3
Let a = relay2      'read the state of relay 2, state will be 0 or 1
```

#### Remote Relay Variables:

Remote relays can be turned on, turned off, pulsed, or toggled in BASIC scripts. Note: remote relays cannot be read (they will always return 0). In order for a remote relay to be used, it must first be set up using the Remote Relay settings in the **Relays** setup tab. The LET statement is used to set the remote relay's state. The states available for remote relays are the same as the internal relays. The relay pulse time is likewise specified in the **Relays** setup tab. The remote relays are designated as rmt\_relay1, rmt\_relay2, rmt\_relay3, etc.

Examples:

```
Let rmt_relay2 = 1  'turn on remote relay 2
Let rmt_relay1 = 0  'turn off remote relay 1
Let rmt_relay2 = 2  'pulse remote relay 2
Let rmt_relay3 = 5  'toggle remote relay 3
```

Note: The remote relay states can only be changed as fast as the remote device can be updated. If the state is changed faster than the device can be updated, only the last state change will occur.

#### Digital Input Variables:

The input states can be read in BASIC scripts. The inputs are designated as input1, input2, input3, etc.

Example:

```
Let a = input1      'sets 'a' equal to value of input 1
```

#### Analog Input Variables:

The analog inputs can be read in BASIC scripts. The inputs are designated as ana1, ana2, ana3, etc.

Example:

```
Let b = ana1        'sets 'b' equal to value of analog input 1
```

---

**Temperature Sensor Variables:**

Each temperature sensor can be read in BASIC scripts. The temperature sensors are designated as temp1, temp2, temp3, etc.

Example:

```

If temp 2 >= 80 Then      'If temperature sensor reads greater than 80
                          'degrees, then
    Let relay2 = 1        'Turn on relay 2
End If
LET c = temp3 + 15       'set user variable 'c' equal to the value on temp3 + 15

```

**Event Variables:**

Actions can be configured to occur at specified days and times. Five events may be configured for the unit. This is done by setting the day and time of the event and then comparing it to the current date and time. Note that the current date and time must be set in the **Date/Time** setup tab. The event date variables are designated ed1 to ed5. They are set using the LET command and setting the desired event date variable equal to the date the event should occur in the format mm/dd/yyyy. The event time variables are designated et1 to et5. They are likewise set using the LET command and setting the corresponding event time variable equal to the event time in the format hh:mm:ss in 24 hour time. The event date variables store the number of days that have passed since January 1, 1970. The event time variables store the number of seconds from the beginning of the day. If the event time variable is incremented more than the number of seconds in a day (86400 seconds), the counter is reset to 0 and the event date variable is incremented by one. The variables 'time' and 'date' are predefined variables that store the current date and time and can only be read. They can then be used to compare the event date and times. The following example demonstrates defining the event variables and comparing it to the current date and time. Assume current time is April 10, 2009 at 1:30 AM and the event should occur in one hour and every hour thereafter.

Example:

```

Let a = 1
Let ed1 = 04/10/2009      'sets the event date to April 10, 2009.
Let et1 = 02:30:00        'sets the event time to 2:30
Do While a <> 0
If ed1 >= date Then       'Tests event date versus current date.
    If et1 >= time Then   'Tests event time versus current time.
        Let et1 = et1 + 3600 'Increments the event time by one hour (in
                            'seconds).
        Print "Event"     'Event to occur
    End If
End If
End If
Loop
End

```

**Examples:**

The following are a few examples of general applications that could be adapted for more specific use.

1. Application: If any one of three sensors exceeds a temperature of 80.5 degrees, relay 1 will be triggered. Note that relay 1 will not turn off until the script is manually restarted.

```

LET relay1 = 0           'Relay 1 is set to the off position
DO                     'The IF THEN statements are looped continuously
  IF temp1 > 80.5 THEN 'Tests temperature sensor 1
    LET relay1 = 1     'Turns on Relay 1
  END IF
  IF temp2 > 80.5 then 'Tests temperature sensor 2
    LET relay1 = 1     'Turns on Relay 1
  END IF
  IF temp3 > 80.5 THEN 'Tests temperature sensor 3
    LET relay1 = 1     'Turns on Relay 1
  END IF
LOOP
END

```

2. Application: If temperature sensor 1 is above 80.5, relay 1 is turned on. If temperature sensor 1 is below 65, relay 2 is turned on. Note that the relays will turn off as soon as conditions return to between 65 and 80.5 degrees.

```

DO                     'The IF THEN statements are looped continuously
  IF temp1 > 80.5 THEN 'Tests if sensor 1 is greater than 80.5 degrees
    LET relay1 = 1     'Turns on Relay 1 if condition is true
  ELSE
    LET relay1 = 0     'Turns off Relay 1 if condition is false
  END IF
  IF temp1 < 65 THEN  'Tests if sensor 1 is less than 65 degrees
    LET relay2 = 1     'Turns on Relay 2 if condition is true
  ELSE
    LET relay2 = 0     'Turns off Relay 2 if condition is false
  END IF
LOOP
END

```

3. Application: Relay 1 is triggered if sensor 1 is above 32 degrees for 10 minutes. It will check every 5 minutes to see if sensor 1 has dropped below 32. If the sensor has dropped below 32 degrees before the 10 minutes are up, the relay will not be triggered. In addition to the relay being triggered, we also want to log the time at which the relay is triggered and send an email notification.

```

DO                     'Loops statements continuously
  DO WHILE temp1 > 32 'Loops IF THEN statements if sensor 1 is greater than
32                     ' degrees
  CALL delay           'Waits 5 minutes
  IF temp1 > 32 THEN  'If sensor 1 is still greater than 32 degrees, then
    CALL delay         'Waits another 5 minutes

```

---

```
        IF temp1 > 32 THEN 'If sensor 1 is still greater than 32 degrees, then
            LET relay1 = 1 'Triggers relay 1
            LOG          'Logs according to selected log settings
            EMAIL temp1 'Sends an email notification from sensor 1
        END IF
    END IF          'If sensor 1 drops below 32 degrees before 10 minutes are up,
    LOOP          '      then the program restarts because the IF statement
LOOP          '      failed
END

SUB delay          'Subroutine delays script by 5 minutes
LET t1 = 3000
DO WHILE t1 <> 0
LOOP
END SUB
```

**Testing and Debugging:**

Note that a copy of the BASIC interpreter for Windows is also available on our website for the use of testing and debugging. The device will only acknowledge errors as it runs. This means that if a path of the script is not encountered, errors may still exist. The Windows version of the interpreter, however, will check for errors before the script is run. Note also that the PRINT function is available for use with the Windows version of the BASIC interpreter. The interpreter on the device, however, will not recognize this command.

Note that the script will be restarted by submitting any of setup pages or by power cycling the unit

## Appendix H: Specifications

### Power Requirements:

#### Voltage Input:

Model X-300-5 : 5VDC  $\pm$  5%

Model X-300-I : 9-28VDC

Model X-300-E : Power Over Ethernet (48V injected into Ethernet line as per 802.3af specification)  
POE Class 1 (0.44Watt to 3.84Watt range) or 5VDC  $\pm$  5%

#### Current Draw (typical):

Network Speed (# of energized relays)	5 VDC (-5 Model)	9 VDC (-I Model)	12 VDC (-I Model)	24 VDC (-I Model)	28 VDC (-I Model)
10 Mbps (No Relays)	0.184 A	0.118 A	0.089 A	0.048 A	0.043 A
10 Mbps (3 Relays)	0.407 A	0.287 A	0.217 A	0.111 A	0.099 A
100 Mbps ( No Relays)	0.314 A	0.201 A	0.151 A	0.081 A	0.071 A
100 Mbps (3 Relays)	0.530 A	0.378 A	0.281 A	0.145 A	0.125 A

\*note that the X-300-E will draw the same current as the X-300-5 when powered with a 5VDC power supply.

### Temperature Sensors:

“1-Wire” Digital Thermometer (Dallas Semiconductor part DS18B20)

Temperature Range:  $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$  ( $-67^{\circ}\text{F}$  to  $+257^{\circ}\text{F}$ )

Accuracy:  $\pm 0.5^{\circ}\text{C}$  from  $-10^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$

Maximum number of Sensors: 8 in *Temperature Monitor* mode 2 in *Thermostat* mode

Maximum Distance from X-300 to Sensors: 120 feet with one sensor. 60 feet with four sensors. Short distances with 8 sensors.

### Temperature Sensor Functions:

Thermometer, Thermostat, Relay Control, Remote Relay Control, Email Alarms, SNMP Traps, Temperature Logging

### Relay Contacts:

Contact Form: SPDT (form c)

Contact Material: AgSnO2

Contact Resistance: 100 m $\Omega$  max.

Max Voltage: 28VAC, 24VDC

Max Current: 3A

Relay Control Options: ON/OFF or Pulsed

Pulse Timer Duration: 100ms to 86400 Seconds (1 day)

### Real-Time Clock:

Time Setup: Manually or with NTP (Network Time Protocol) server

NTP Synchronization Period: User specified (once, daily, weekly, monthly, and/or on power-up)

Automatic Daylight Savings Adjustment: Yes

Battery Backup: Internal capacitor backup maintains time for several days

### Logging:

Number of Log Files: 2 (System log and Data log)

Log Storage: Nonvolatile Flash Memory

Buffer Architecture: Circular Buffer

Data Logging Type: Selectable Event and/or Periodic

System Logging Type: Event

Data Log File Size: 200K bytes (approx 4655 logs)

System Log File Size: 16K bytes

---

**Operating Modes:**

Temperature Monitor - used for temperature monitoring, alarming, discrete relay control, advanced functions.

Thermostat - advanced single-stage HVAC system controller.

**Advanced Features:**

BASIC interpreter

Remote services

**Network:** 10/100 Base-T Ethernet

**Network Setup:** Static IP address assignment or DHCP, HTTP port selectable

**Protocols Supported:** HTTP, Modbus/TCP, SNMP, SMTP, plus proprietary Remote Services

**Internal Monitoring:** Vin Voltage, 5V, internal temperature

**Connectors:**

Power/Sensors/Relays: 14-position, removable terminal strip, 3.81mm terminal spacing  
(Replacement part number, Phoenix Contact 1803691)

Network: 8-pin RJ-45 socket

**LED Indicators: 6**

- Power Indicator
- Relay coil engaged (Relay 1, Relay 2, Relay 3)
- Network linked
- Network activity

**Physical:**

Operating Temperature: -30° to 65°C ( -22°-149°F)

Size: 1.41in (35.7mm) wide X 3.88in (98.5mm) tall X 3.1 in(78.0mm) deep

Weight: 5.5oz (156 grams)

Enclosure Material: Lexan 940 (UL94 V0 flame rated)

**Password Settings:**

Password protection on setup page: Yes

Password protection on control page: Optional

Password Encoding: Base 64

Max setup password length: 13 characters

Max control password length: 13 characters

**Regulatory Compliance:**

Electromagnetic Compliance:

IEC CISPR 22, CISPR 24

FCC 47CFR15 (-I and -5 Models class B, POE Model Class A)

EU EN55024, EN55022

Product Safety:

IEC 60950-1 / EN 60950-1



### Appendix I: Mechanical Information

