7. XPRESS Commands

This chapter fully describes all XPRESS commands, how they are used, and what they require for hardware. Note also that some work only in the sequential control section.

AC power failed since the last test — ACFAILED

Indicates that the AC power has failed since the last time this keyword was tested. Power may or may not be back on when this is true. When using a SpectraSense, the result reflects AC power activity within the last second or so. When using a PL-Link, the result reflects AC power activity during the last minute. ACfailed is cleared to false right after being tested. See ACpower for more details.

Syntax: ACfailed

Example:
```
IF ACfailed=TRUE and ACpower=ON THEN
  Say"The power failed, but has come back on"
END
```

Status of AC power — ACPOWER

In battery-backed systems, the Supervisory Controller continues operating even when AC power has failed. It is often useful to know when the power has gone out so appropriate action can be taken either during the outage or to reinitialize devices in the house when power returns.

ACpower gets its information from either the PLIX chip in a SpectraSense system or the PL-Link otherwise. The result reflects the state of the AC power immediately when using the PLIX (since the chip is right on the processor board), but the PL-Link is polled for the AC power status only once per minute. As a result, the AC power may be out for up to one minute before the Supervisory Controller knows about it. Use ACfailed to determine if the power went out and came back on in less than a minute.

Syntax: ACpower

Example:
```
IF ACpower=ON THEN
  Say"The power is currently on."
ELSE
  Say"Power has been lost."
END
```

ADC Test analog channel (local and network)

Any analog channel in the system (including local and network) may be tested. See Appendix B for tables that assign channel numbers to specific hardware. Valid value ranges depend on the resolution of the A/D converter being used. For an 8-bit part, the range is 0–255; for 10 bits, it's 0–1023; and for 12 bits, it's 0–4095.

Syntax: ADC(n)

where: n = valid ADC channel (0–127)

Hardware: ADIO (optional), DIO+ (optional)

Example:
```
IF ADC(16)>=Variable(10) THEN
  Module(L6) = ON
END
```

ALLLIGHTSON Turn on all X-10 lights in a housecode

X-10 modules support an "All Lights On" command that turns on just lamp and wall-switch modules set for the same housecode. This command is ignored by all other X-10 modules (including appliance modules) regardless of what is plugged into them.

Syntax: AllLightsOn(h)

where: h = housecode (A–P)

Hardware: PL

Example:
```
IF Input(8)=ON THEN
  AllLightsOn(B)
END
```

ALLUNITSOFF Turn off all X-10 modules in a housecode

X-10 modules support an "All Units Off" that turns off all X-10 modules set for the same housecode.

Syntax: AllUnitsOff(h)

where: h = housecode (A–P)

Hardware: PL

Example:
```
IF Timer(18)>=30 THEN
  AllUnitsOff(G)
END
```

Turn Caller ID support on and off — CALLERID

When a modem that supports Caller ID is connected to the HCS/SS, the CallerID command turns the support on and off. Do not attempt to use this command if your modem doesn't explicitly support Caller ID. Using this command when there is no modem connected has no effect on the system. The Caller ID in Practical Peripherals modems is *not* supported. See Section 5.3 for more details on compatible modems.

Syntax: CallerID = c

where: c = ON/OFF/TRUE/FALSE

Hardware: External Caller ID modem

Example:
```
IF Reset THEN
        CallerID = ON
      END
```

Determine progress of current phone call — CALLPROGRESS

Once a call has been dialed, the system must know what is happening on the line to know how to proceed. The CallProgress function accepts a number that tells it the maximum number of rings to wait. It returns the final status of the call: no answer, busy , or answered.

Be sure to use CallProgress just once for a particular outgoing call. To test its result using multiple IF statements, assign the result to a variable and test the variable instead.

Only works in the sequential section.

Syntax: r = CallProgress(n)

where:	n = max number of rings (1–255)
	r = 0: no answer after n rings
	1: busy
	2: answer

Example:

```
DEFINE CP = Variable(10)
!
! Pick up the phone, check for dial tone,
!  then dial a number, wait for up to
!  three rings, and deliver a message if
!  answered
!
IF Input(3)=TRUE THEN
  OffHook
  IFA DialTone=TRUE THEN
    DialStr("5551212")
    CP = CallProgress(3)
    IFA CP=0 THEN
      LCD(0) = "No answer\n"
    END
    IFA CP=1 THEN
      LCD(0) = "Busy\n"
    END
    IFA CP=2 THEN
      SayW"This is an automated message"
    END
  END
  OnHook
END
```

CID*xxxx* Various Caller ID information

When a modem that supports Caller ID is connected to the HCS/SS and the phone rings, the Caller ID data is passed to the HCS/SS between the first and second rings. That data is presented to your XPRESS program using individual keywords described below.

CIDnew = TRUE for new data, FALSE after one pass through program
CIDmonth = month call received (1–12)
CIDday = day of month call received (1–31)
CIDhour = hour call received (0–23)
CIDminute = minute call received (0–59)
CIDarea = area code of calling number (000–999)

CIDexch = exchange of calling number (000–999)
CIDnumber = last four digits of calling number (0000–9999)

Out-of-area calls return 000 in CIDarea and CIDexch, and 0000 in CIDnumber. Private (blocked) calls return 999 in CIDarea and CIDexch, and 9999 in CIDnumber.

Hardware: External Caller ID modem

Example:

```
DEFINE Known = Variable(0)

BEGIN

IF Reset THEN
  CallerID = ON
END

IF CIDnew=TRUE THEN
  Known = FALSE
  IFA CIDarea=000 THEN
    Say"Out of area"; Known=TRUE
  END
  IFA CIDarea=999 THEN
    Say"Private number"; Known=TRUE
  END
  IFA CIDarea=203 THEN
    IFA CIDexch=871 THEN
      IFA CIDnumber=1271 OR CIDnumber=6170 OR
             CIDnumber=6866 THEN
        Say"Micromint"; Known=TRUE
      END
      IFA Known=FALSE THEN
        Say"Unknown Vernon"; Known=TRUE
      END
    END
    IFA CIDexch=875 THEN
      IFA CIDnumber=2199 OR CIDnumber=2751 THEN
        Say"Circuit Cellar"; Known=TRUE
      END
      IFA CIDnumber=5795 THEN
        Say"Micromint"; Known=TRUE
      END
      IFA Known=FALSE THEN
        Say"Unknown Vernon"; Known=TRUE
      END
    END
  IFA Known=FALSE THEN
    Say"Unknown in state"; Known=TRUE
  END
 END
 IF Known=FALSE THEN
  Say"Unknown out of state"
 END
END
```

CLEARLOG Clear all logged data values from memory

On a system reset or new program load, the logged-data pointer is left untouched (and uninitialized if starting the system from a cold start). Before any data logging is done, the data pointer should be cleared either with the HOST “C” command (in the Log menu) or with this command at the top of your XPRESS program.

Syntax: ClearLog

Example:

```
IF Reset THEN
  ClearVariables; ClearTimers
  ClearLog
END
```

CLEARTIMERS Stop and clear all system timers

On a system reset or new program load, all system timers are left running (if they were running before the reset or program load). ClearTimers allows you to stop and clear all the system timers with a single command.

Syntax: ClearTimers

Example:

```
IF Reset THEN
  ClearVariables; ClearTimers
  ClearLog
END
```

Clear all system variables to 0 — CLEARVARIABLES

On a system reset or new program load, all system variables are left with their previous values (or undefined). ClearVariables allows you to set all the system variables to 0 with a single command.

Syntax: ClearVariables

Example:
```
IF Reset THEN
  ClearVariables; ClearTimers
  ClearLog
END
```

Define what hardware is being used with the HCS/SS — CONFIG

Every HCS/SS installation contains a Supervisory Controller (SC). The SC may also have up to two BUFIO or BUF50 expansions boards on top to provide more local I/O bits. A number of COMM-Link modules are also connected to the SC in most installations. The SC must be told what hardware is in the system so it knows what I/O bits to display and what network modules to poll.

If the SC is expecting to find a network module that isn't connected, the systems continue to run, but performance degrades. To define the SC hardware and how many of each network module is connected, CONFIG is used. All CONFIG statements must precede the BEGIN statement. (Note that only one PL-Link may be used on the network.

BUF-Term boards should *not* be defined.

Setting a configuration that doesn't match the hardware available won't magically make new features appear. For example, if you have an HCS2-DX board with an 8-bit ADC on it, setting ADCRES to 12 won't give you any extra bits of resolution. In fact, it will likely result in strange and inaccurate readings from the ADC. Likewise, ADCGAIN only works with a Spectra-Sense board since only that board has special gain-setting circuitry on it.

All COMM-Link modules must start with network address 0 and be assigned consecutive addresses.

Syntax:

```
Config <module> = m
Config SC = <processor>
Config BUF = b
Config IND54 = io,io,io,io
Config ADCRES = r
Config ADCGAIN = g0,g1,g2,g3,g4,g5,g6,g7
```

where:

<module> =	"PL-Link"	
	"IR-Link"	(use with MCIR-Link also)
	"LCD-Link"	
	"DIO-Link"	
	"ADIO-Link"	
	"DIO+-Link"	

m = number of network modules connected (0–7)

<processor> =	"HCS180" (original SC and newer DX)
	"IND180" (bus-based processor)
	"SpectraSense"

b = number of BUFIO or BUF50 boards in system (1 or 2; do not count any BUF-Term boards)

io = IN or OUT indicating the direction of each 8255 on the board (there must be 4 parameters)

r = ADC resolution (only 8, 10, and 12 are valid)

gx = gain for each ADC channel (only 1, 2, 4, and 8 are valid; there must be 8 parameters)

Example:

```
!
! Original HCS II with BUF-Term and BUFIO installed and
!  three LCD-Links on the network
!
Config SC = HCS180
Config BUF = 1
Config LCD-Link = 3
Config ADCRES = 8
!
! IND180 system with IND54 and IND30 installed and
!  one PL-Link on the network
!
Config SC = IND180
Config IND54 = IN,IN,IN,OUT
Config ADCRES = 12
Config PL-Link = 1
!
```

```
! SpectraSense 2000 board with BUFIO and 10-bit ADC
!  installed
!
Config SC = SpectraSense
Config BUF = 1
Config ADCRES = 10
Config ADCGAIN = 1,1,1,1,4,2,8,2
```

Send a message to a window on the console — CONSOLE

While debugging a system, it is often useful to display status messages from within the XPRESS program to give the installer an idea of what's going on. Text strings containing optional variable and time/date information may be sent to the console.

Important: No ANSI processing is done on messages sent to the screen! Characters are displayed exactly as sent to the console. There is no way to control the on-screen cursor as there is with the LCD-Link. A carriage return is added to the end of each string displayed.

Syntax: `Console = "<string>", Variable(x), Variable(x), …`

where: <string> = text string with embedded time, date, and variable descriptors

See the LCD command in this section for complete details of the descriptors and how they are used.

Example:

```
DEFINE Reading = Variable(0)
BEGIN
IF Input(4)=ON THEN
  Reading = (ADC(0) * 50) / 256
  Console = "Value = %P0", Reading
END
```

Set a network analog channel — DAC

Set any DAC channel on any network ADIO module to an 8-bit value. See Appendix B for a list of channel numbers and corresponding ADIO module numbers.

Syntax: DAC(n)

where: n = channel number (0–31)

Hardware: ADIO

Example:
```
IF ADC(16)>=192 THEN
  DAC(2) = 32
END
```

DAY Current day of month

Retrieve the day of the month. Day will be in the range of 1–31.

Syntax: Day

Examples:
```
IF Day=1 THEN
  LCD(0) = "Flip the calendar\n"
END
```

The Day operand may be used in conjunction with a variable and an equation to determine whether the current day of the month is odd or even. Taking advantage of the integer math, the Odd calculation below results in a 0 (false) whenever Day is even and a 1 (true) whenever Day is odd.

```
DEFINE Odd = Variable(2)
DEFINE Sprinkler = Module(F2)

BEGIN
IF Time=8:00:DY THEN
  Odd = Day – ((Day / 2) * 2)
  IFA Odd=TRUE THEN
    Sprinkler = ON
    LCD(0) = "Sprinkler day\n"
  ELSE
    LCD(0) = "No sprinkling today\n"
  END
END
```

Decrement a variable — DEC

Variables may contain any 16-bit number and may be decremented to be used as counters.

Syntax: Dec(Variable(n))

where: n = variable number (0–127)

Example:

```
DEFINE Counter = Variable(2)

IF Input(4)=EDGE AND Input(4)=ON THEN
  Dec(Counter)
END
```

Define a descriptive program label — DEFINE

Any valid XPRESS expression or action may be assigned to a descriptive label to enhance program readability. DEFINE is used to equate the label with the expression or action. All DEFINE statements must precede the BEGIN statement. Up to 512 labels may be defined with each label name up to 32 characters long. Label names may not contain embedded spaces.

Syntax: Define <label> = <statement>

where: <label> = any alphanumeric string up to 32 characters long. No spaces are allowed.
<statement> = any valid XPRESS expression or action

Example:

```
DEFINE KitchenLight  = Module(L4)
DEFINE KitchenMotion = Input(2)
BEGIN
IF KitchenMotion = ON THEN
  KitchenLight = ON
END
```

DIALDIGIT Dial a single DTMF digit contained in the given variable

The variable must contain a value between 0 and 15 (corresponding to all ten numbers, star, pound, and four letter DTMF symbols). Upon calling this function, the HCS-DTMF board or SpectraSense will dial the single digit. The phone must already be off hook (using the OffHook command).

Only works in the sequential section.

Syntax: DialDigit(Variable(n))

where: n = variable number containing the digit to be dialed (0–9, 10 [*], 11 [#], 12–15 [A–D])

Hardware: HCS-DTMF or SpectraSense

Example:

```
DEFINE Digit = Variable(8)
!
SEQUENTIAL
IF Rings>=2 THEN
  OffHook
  Say"Press a button"
  Say"and I will repeat it back to you."
  Digit = DTMFdigit(150)
  Wait(5)          ! Pause half a second
  DialDigit(Digit)
  OnHook
END
```

DIALNUMBER Dial up to four DTMF numbers from the given variable

The variable must contain a value between 0 and 9999 (corresponding to one to four DTMF numbers). Upon calling this function, the HCS-DTMF board or SpectraSense will dial as many digits as specified by the second parameter. If there are more digits in the variable than are requested, the right-most digits are used. For example, if the variable contains the number 5678 and the DialNumber function is called with a 3 in the second parameter, the numbers 6, 7, and 8 are dialed (the 5 is discarded). The phone must already be off hook (using the OffHook command).

Only works in the sequential section.

Syntax: DialNumber(Variable(n),d)

where: n = variable number containing the numbers to be dialed (0–9999)
d = number of digits to be dialed (1–4)

Hardware: HCS-DTMF or SpectraSense

Example:

```
DEFINE Num1 = Variable(10)
DEFINE Num2 = Variable(11)
!
! After three rings, pick up phone and
!  accept a seven-digit phone number.
!  Then hang up the phone, call that
!  number back, and deliver a message.
!
SEQUENTIAL
IF Rings>=3 THEN
  OffHook
  Say"Enter a seven digit phone number,"
  Say"followed by the pound sign."
  Num1 = DTMFnumber(150)
  IFA Num1>=0 THEN
    Num2 = DTMFnumber(50)
  END
  IFA Num1<0 OR Num2<0 THEN
    LCD(0)="Timed out!\n"
  ELSE
    LCD(0)="Number = %P0%P0\n",Num1,Num2
    OnHook
    Wait(20)                          ! 2 seconds
    OffHook
    IFA DialTone=TRUE THEN
      DialNumber(Num1,4)
      DialNumber(Num2,3)
      CP = CallProgress(3)
      IFA CP=0 THEN
        LCD(0) = "No answer\n"
      END
      IFA CP=1 THEN
        LCD(0) = "Busy\n"
      END
      IFA CP=2 THEN
        Say"This is your callback."
        SayW"Good bye."
      END
    END
  END
  OnHook
END
```

DIALSTR Dial the sequence in the given text string

All ten numbers, star, pound, and four letter DTMF symbols may be dialed using the HCS-DTMF board or SpectraSense. The phone must already be off hook (using the OffHook command). A comma embedded in the string generates a 2-second pause.

Only works in the sequential section.

Syntax: DialStr(“string”)

where: string = text string containing digits to be dialed
(0–9,*,#,A–D, and comma)

Hardware: HCS-DTMF or SpectraSense

Example:

```
DEFINE Alarm = Input(5)
!
! On an alarm condition, call a
!  preprogrammed number and
!  report the trouble.
!
SEQUENTIAL

IF Alarm=TRUE THEN
  OffHook
  DialStr("5551212")
  Wait(100)                          ! 10 seconds
  SayW"Alarm condition detected."
  Wait(50)                           ! 5 seconds
  OnHook
END
```

Listen to phone line and wait for dial tone DIALTONE

It’s best to make sure you have a working phone line before trying to make a call. DialTone will use the HCS-DTMF board or SpectraSense to listen to the phone line for two seconds and if it detects a steady tone, will return a TRUE result. If it doesn’t detect a tone, it will return FALSE.

Only works in the sequential section.

Syntax: DialTone

Hardware: HCS-DTMF or SpectraSense

Example:

```
DEFINE CP = Variable(10)
!
! Pick up the phone, check for dial tone,
!  then dial a number, wait for up to
!  three rings, and deliver a message if
!  answered
!
IF Input(3)=TRUE THEN
  OffHook
  IFA DialTone=TRUE THEN
    DialStr("5551212")
    CP = CallProgress(3)
    IFA CP=0 THEN
      LCD(0) = "No answer\n"
    END
    IFA CP=1 THEN
      LCD(0) = "Busy\n"
    END
    IFA CP=2 THEN
     SayW"This is an automated message"
    END
  END
  OnHook
END
```

DISPLAY Define which X-10 housecodes to display by HOST

You may specify which X-10 housecodes you want displayed by the HOST program. If this line is omitted, no housecodes are displayed.

Syntax: Display Modules = <housecodes>

where: <housecodes> = list of housecode letters separated by commas

Example: DISPLAY Modules = A,B,C,F

DTMFDIGIT Receive a single DTMF digit

All ten numbers, the star and pound sign, and the four letter DTMF symbols may be received from the phone line. The phone must already be off hook (using the OffHook command). The result may be assigned to a variable or used anywhere an math expression may be used. A timeout from 0.1 to 25.5 seconds may be set to avoid hanging the system. Sequential XPRESS processing is suspended until a DTMF digit is received or the command times out. When a timeout occurs, the function returns a –1.

Only works in the sequential section.

Syntax: d = DTMFdigit(t)

where: d = DTMF symbol (0–15)
(0–9, 10 [*], 11 [#], 12–15 [A–D])
(–1 if timeout occurs)
t = timeout value in tenths of seconds (1–255)
(0.1–25.5 seconds)

Hardware: HCS-DTMF or SpectraSense

Example:

```
DEFINE Num = Variable(10)
!
SEQUENTIAL

IF Rings>=2 THEN
  OffHook
  Say"Please select a function."
  Num = DTMFdigit(150)
  IFA Num=1 THEN
     .
     .
  END
  IFA Num=2 THEN
    .
    .
  END
  SayW"Thank you. Goodbye."
  OnHook
END
```

Receive up to four DTMF digits — DTMFNUMBER

Up to four DTMF number symbols may be received and combined to make up a single 16-bit value (0–9999). The command returns when either four digits have been received or the star or pound button is pressed (indicating fewer than four digits have been entered). The phone must already be off hook (using the OffHook command).

The result may be assigned to a variable or used anywhere a math expression may be used. A timeout from 0.1 to 25.5 seconds may be set to avoid hanging the system. The timeout timer is reset for each button press (so for a timeout value of 100, the caller has 10 seconds per button press, not 10 seconds total). Sequential XPRESS processing is suspended until a complete number is assembled or the command times out. When a timeout occurs, the function returns a –1.

Only works in the sequential section.

Syntax: n = DTMFnumber(t)

where: n = number made up of DTMF numbers (0–9999)
(* and # terminate number; A–D are ignored)
(–1 if timeout occurs)
t = timeout value in tenths of seconds (1–255)
(0.1–25.5 seconds)

Hardware: HCS-DTMF or SpectraSense

Example:

```
DEFINE Num1 = Variable(10)
DEFINE Num2 = Variable(11)
!
SEQUENTIAL

IF Rings>=3 THEN
  OffHook
  Say"Enter a seven digit phone number,"
  Say"followed by the pound sign."
  Num1 = DTMFnumber(150)
  IFA Num1>=0 THEN
    Num2 = DTMFnumber(100)
  END
  IFA Num1<0 OR Num2<0 THEN
    LCD(0)="Timed out!\n"
  ELSE
    LCD(0)="Number = %P0%P0\n",Num1,Num2
  END
  OnHook
END
```

```
DEFINE PW = Variable(10)
!
SEQUENTIAL

IF Rings>=3 THEN
  OffHook
  Say"Enter your password."
  PW = DTMFnumber(150)
  IFA PW=5432 THEN
    SayW"Correct. Thank you."
  ELSE
    SayW"Wrong. Goodbye."
  END
  OnHook
END
```

DOW Current day of week

Retrieve the current day of the week. Day-of-week tests may also be done with the Time keyword. DOW is 1 on Sunday and 7 on Saturday.

Syntax: `DOW`

Example:

```
IF DOW=3 THEN
  LCD(0) = "Today is Tuesday\n"
END
```

Current hour — HOUR

Retrieve the current hour of the day. Hour will be in the range of 0–23.

Syntax: Hour

Examples:

```
IF Input(8)=ON THEN
  Variable(5) = Hour
  Variable(6) = Minute
  Variable(7) = Second
END
```

Increment a variable — INC

Variables may contain any 16-bit number and may be incremented to be used as counters.

Syntax: Inc(Variable(n))

where: n = variable number (0–127)

Example:

```
DEFINE Counter = Variable(2)

IF Input(4)=EDGE AND Input(4)=ON THEN
  Inc(Counter)
END
```

Test or set a local digital input — INPUT

All local digital input numbers are predefined and depend on the hardware being used. See Appendix B for a table of supported hardware and corresponding input numbers.

Syntax: Input(n) = ON/OFF/EDGE

where: n = input number (0–207)

Example:

```
IF Input(5)=EDGE THEN
  Output(8)=ON
END
```

Inputs are normally either on or off. The transition from on to off or from off to on is known as an “edge.” Detecting edges alone (as opposed to either on or off) is useful when you want to know that the state of an input has changed, but don’t care whether it changed to on or off.

Edge detection is done in software and requires a pulse width of at least 250 ms to detect local input edges (netbit edges require more time). Rising edges (off to on) and falling edges (on to off) are treated identically. You may determine whether a rising or falling edge has occurred by testing the input’s level at the same time (e.g., IF Input(3) = EDGE AND Input(3) = ON THEN it was a rising edge).

Edges may be tested multiple times during a single pass through the XPRESS program. At the end of each pass through the list, the edge storage table is cleared.

IRCODE Test whether an IR code has been received

Up to eight MCIR-Link modules may be connected to the network, any of which can receive codes from hand-held IR remotes. IRCODE is used to test whether a particular code has been received and which MCIR-Link module received it.

Syntax: IRcode(c) <op> <val>

where: c = IR code (0–239)
<val> refers to an IR-Link module number, so really only makes sense when in the range of 0–7.

Hardware: MCIR

Examples:

```
! If any MCIR-Link module has received IR code number 6,
! the X-10 module G6 is turned off.
!
IF IRcode(6)>=0 THEN
  Module(G6)=OFF
END

! If MCIR-Link number 3 has received IR code number 12,
! output 5 will be turned on.
!
IF IRcode(12)=3 THEN
  Output(5) = ON
END
```

```
! Suppose MCIR-Link number 1 is located in the family room
! and IR code number 5 is defined as putting the system into
! “movie mode” (i.e., “I’m sitting down to watch a movie, so set
! up the room appropriately.”). When you press the button on
! your hand-held IR remote to send IR code number 5, HCS/SS
! dims two lights and sets an output that triggers the automated
! drapes to close.
!
DEFINE MovieMode  = IRcode(5)
DEFINE FamilyRoom = 1
BEGIN
IF MovieMode=FamilyRoom THEN
  Module(D4) = Dim(8); Module(D5) = Dim(10)
  Output(12) = ON
END
```

Send a message to an LCD-Link — LCD

A message may be sent to any LCD-Link on the network consisting of text, current time and date, and variable values. The subset of ANSI control sequences supported by the LCD-Link may be used to format the display. Lines too long for the display are wrapped to the next line. When the bottom of the display is reached, the display will scroll. See the LCD-Link documentation for more information about the supported ANSI commands.

Syntax: LCD(n) = “string”, Variable(x), Variable(y), …

where: n = LCD-Link number (0–7)
string = text string with embedded time, date, and variable descriptors

Hardware: LCD

Control characters within the string follow the conventions outlined in the LCD-Link manual. Within “string” may be time, date, and variable descriptors. The descriptors are as follows:

%A = time (HH:MM)
%B = time (HH:MM:SS)
%C = day of week (Sun, Mon, …, Sat)
%D = date (MM/DD/YY)
%E = date (Month DD, YYYY; Month = Jan, Feb, …, Dec)

To display variables, a format descriptor is used within the string and the variable to be displayed is appended to the end of the string. The following formats are supported:

> %Px = print a variable with *x* digits after the decimal point and no leading zeros or spaces
> %Qx = print a variable with *x* digits after the decimal point and leading zeros replaced with spaces
> %Rx = print a variable with *x* digits after the decimal point and leading zeros

In the case of %P, only as many characters as necessary to represent the number are printed. In the case of %Q and %R, five digits are always printed (though leading zeros are replaced by spaces with %Q), with the decimal point taking up one more character position. A leading negative sign is added to any negative number. When *x* is zero, the decimal point isn't displayed at all.

Examples:

```
LCD(0) = "\e[2JToday is %D\n  or\n%C, %E\n"
```

This statement clears the display on LCD-Link 0 and creates the following message:

```
Today  is  03/15/95
  or
Wed,  Mar  15,  1995
```

In the following excerpt, when the button connected to input 5 is pressed to request a reading, the ADC is read and its raw value is assigned to a variable. The raw reading (0–255) is converted to the actual voltage being measured (0–5 volts) and both are displayed on the LCD-Link display. Note that variable 15 contains a value ten times larger than the actual voltage. When displayed with one digit after the decimal point, it shows the correct value and simulates floating point, though only integer math is involved.

```
DEFINE RawV = Variable(9)
DEFINE DisplayReading = Input(5)

BEGIN
IF DisplayReading=ON THEN
  RawV = ADC(2)
  Variable(15) = (RawV * 50) / 256
  LCD(0) = "Reading = %P1 volts\n",
          Variable(9)
  LCD(0) = "Raw value = %P0 \n", RawV
  LCD(0) = "Raw value = %Q0\nRaw value = %R0",
          RawV, RawV
END
```

```
Reading   =   3.1    volts
Raw  value  =  161
Raw  value  =      161
Raw  value  =  00161
```

Tip: To display a degree symbol ("°") on an LCD-Link, use the sequence "\xDF" in your display string. For example, the string "Temp = 28\xDF" displays "Temp = 28°" on the LCD-Link.

Log a data value to memory LOG

At any time, a 16-bit value may be saved to memory for later processing by the user. The time and date of logging is saved with the value. A reference ID number is also stored with the value so different sensors and events may be logged to the same memory and sorted later. Each log entry uses eight bytes.

You must have an extra 32K SRAM installed in socket U10 of the HCS180 or HCS2-DX, socket U11 of the IND180, or socket U14 of the SpectraSense in order to log data. See Chapter 8 for details on how to install an extra RAM chip and how the logged data is formatted.

Syntax: Log(i) = <val>

where: i = reference ID number (0–254)

Example:

```
!
! Log the raw outside temperature reading
!  every 30 minutes and log whenever the
!  motion sensor is activated and whenever
!  the front door is opened.
!
DEFINE Temp = 0
DEFINE Motion = 1
DEFINE FrontDoor = 2

BEGIN
IF Reset THEN
  ClearLog
  Timer(70) = ON
END
```

```
IF Timer(70)>=30 THEN
  Log(Temp) = ADC(2)
  Timer(70) = ON
END
IF Input(2)=ON THEN
  Log(Motion) = 0
END
IF Input(5)=ON THEN
  Log(FrontDoor)
END
```

Note that when logging the motion and front door, the actual value saved isn't important. Saving the reference ID and current time and date is enough to tell you when each sensor was tripped. After the raw data has been dumped to a file on the PC, a program may be written to sort the data by reference IDs and produce a final report with data grouped by temperatures, motion, and front door activity.

LOGSIZE Check the size of logged data

When the 32K log buffer becomes full, the HCS/SS starts throwing away the oldest samples to make room for new samples. Using the LogSize keyword, it's possible to have the system alert the user that the log memory is close to filling up. The user can then use HOST to dump the logged data to a disk file on a PC. LogSize returns a value in the range of 0–4095 representing the number of logged entries. Note that each log entry takes up eight bytes.

Syntax: LogSize

Example:
```
IF LogSize>4000 THEN
  LCD(0) = "Time to empty the buffer.\n"
END
```

LPT Send a message to a DIO-Link

A message may be sent to any DIO-Link on the network that has a byte-wide device connected to it (such as a printer) consisting of text, current time and date, and variable values. The subset of ANSI control sequences supported by the DIO-Link may be used to format the output. See the DIO-Link documentation for more information about device connection and supported ANSI commands.

Syntax: LPT(n) = “string”, Variable(x), Variable(y), …

where: n = DIO-Link number (0–7)
string = text string with embedded time, date, and variable descriptors

Hardware: DIO

See the LCD command in this section for complete details of the descriptors and how they are used.

Tell an MCIR-Link to send a trained infrared command — MCIR

The MCIR-Link stores trained IR commands and references them by number. You must set up (train) the MCIR-Link in interactive mode before connecting it to the network. XPRESS only allows you to instruct the MCIR-Link to send a command that has already been stored. It does not allow remote training or loading of the MCIR-Link module. See the MCIR-Link manual for more details.

Syntax: MCIR(n) = <val>

where: n = MCIR-Link number (0–7)
<val> must be in the range of 1–999 or 1001–1999

Hardware: MCIR

Example:
```
DEFINE TVon = 6      ! Trained MCIR code
BEGIN
IF Input(8)=ON THEN
  LCD(0) = "Turning on TV"
  MCIR(0) = TVon
END
```

Current minute — MINUTE

Retrieve the current minute. Minute is in the range of 0–59.

Syntax: Minute

Example:
```
IF Input(8)=ON THEN
  Variable(5) = Hour
  Variable(6) = Minute
  Variable(7) = Second
END
```

MODEMINIT User-defined modem initialization string

When a modem is connected to the HCS/SS, the default initialization string "AT&FMV&D&K&Q5" is used to set up the modem. This string should work for virtually all modems, so the ModemInit command should not be used unless absolutely necessary. Be sure you know exactly what you're changing or the modem may react in a way that the HCS/SS doesn't expect and can't handle. Always include the "AT" attention code at the start of the string.

Syntax: ModemInit = "string"

where: string = valid modem commands preceded by "AT"

Hardware: External modem

Example:
```
IF Reset THEN
  ModemInit = "ATE1Q&C"
END
```

MODEMRINGS Number of rings to wait before answering modem

When a modem is connected to the HCS/SS, ModemRings defines how many rings the modem will wait before answering the phone. Only a PC with a modem running HOST should be used to call the HCS/SS.

Syntax: ModemRings = r

where: r = constant (0–9), Variable(n), Equation
(use 0 or OFF to disable modem answer)

Hardware: External modem

Example:
```
IF Reset THEN
  ModemRings = 2
END
```

Test the current state of an X-10 module — MODULE

X-10 modules may only be tested for on or off. Receiving all dim and bright commands with the PL-Link or SpectraSense is impossible, so only a module's on or off state is recorded. Determining the proper dim or bright level for a given situation is up to the user.

Syntax: Module(hnn) = ON/OFF

where: h = house code (A–P)
nn = module number (1–16)

Hardware: PL or SpectraSense

Example:
```
IF Module(B8)=ON AND Input(7)=EDGE THEN
  Output(8)=OFF
END
```

Send out an X-10 command — MODULE

Turn any kind of X-10 module on or off. Lamp and switch modules may also be dimmed or brightened. Note that an X-10 module that is off must go to full brightness before its level may be adjusted. A module may not be gradually brightened from black unless it was previously turned full on and dimmed to black (*not* turned off).

Syntax:
```
Module(hmm) = ON/OFF/ONA
Module(hmm) = DIM(n)
Module(hmm) = BRIGHT(n)
```

where: h = housecode (A–P)
mm = module number (1–16)
n = number of dim/bright steps (1–48)

Hardware: PL or SpectraSense

When the Supervisory Controller knows that a particular X-10 module is already on, it won't send another ON command to it. To force the SC to always send an ON command, even if the module is already on, use the ONA (ON Always) command.

Examples:
```
IF Input(3)=ON THEN
  Module(J9) = Dim(6)
END
```

```
! Module C1 is a chime module, which
!  sounds its chime any time it receives
!  an ON command. There is no need to
!  send it an OFF command because the
!  module just ignores it. We use ONA to
!  force the system to send an ON command
!  even though it thinks the module is
!  already on.
!
IF Input(5)=ON THEN
  Module(C1) = ONA
END
```

MONTH Test current month

Retrieve the current month. Month is 1 in January and 12 in December.

Syntax: Month

Example:

```
IF Month=10 THEN
  LCD(0) = "The month is October\n"
END
```

NETBIT Test or set a network I/O bit

Network I/O ports on DIO-Link modules may be either inputs or outputs depending on how they are referenced. If a bit is set to 1 or 0, its output will go high or low and is treated as an output. If a bit is set to 1 and externally driven, it is considered an input. The direction of I/O bits on all other COMM-Links is fixed (as described in Appendix B). The current state of a bit, whether input or output, may be tested. Fixed bit numbers are assigned to COMM-Link modules as described in the table in Appendix B.

Syntax: Netbit(n) = ON/OFF/EDGE

where: n = valid netbit number (0–319)

Hardware: DIO, ADIO, LCD, DIO+

Examples:

```
IF Netbit(3)=ON THEN
  Module(L5) = ON
END
```

```
IF Input(23)=ON THEN
  Netbit(7) = OFF
END
```

Netbit inputs are normally either on or off. The transition from on to off or from off to on is known as an “edge.” Detecting edges alone (as opposed to either on or off) is useful when you want to know that the state of an input has changed, but don’t care whether it changed to on or off.

Edge detection is done in software and requires a pulse width of at least 1 second and (for complex networks) sometimes up to 5 seconds to detect netbit edges. Rising edges (off to on) and falling edges (on to off) are treated identically. You may determine whether a rising or falling edge has occurred by testing the input’s level at the same time (e.g., IF Input(3) = EDGE AND Input(3) = ON THEN it was a rising edge).

Edges may be tested multiple times during a single pass through the XPRESS program. At the end of each pass through the list, the edge storage table is cleared.

Manage netbits eight at a time — NETBYTE

Eight netbits may be managed simultaneously by handling them as a single byte. All 40 netbytes may be read and tested, but only the first 8 (corresponding to the DIO-Links) may be set. See Appendix B for a listing of netbyte numbers.

Syntax: Netbyte(b)

where: b = netbyte number (0–39)

Examples:

```
IF Input(8)=ON THEN
  Netbyte(3) = 78
END

IF Netbyte(8)>0 THEN
  LCD(0) = "Button pressed\n"
END
```

NETWORK Send a raw text string to the network

A raw network-module command may be sent directly from XPRESS to the network. Strings to be sent to the network should start with the address of the network module, followed by any commands for the module (e.g., TERM0 S=Testing). The HCS/SS automatically inserts the proper lead-in character and checksum. No command may be sent to a network module that produces a reply from the module!

Use extreme caution when sending strings to the network! No checking is done by the SC to determine that what is being sent is valid. It’s possible to cause glitches on the network with unwanted replies generated in response to strings sent to network modules.

Syntax: Network = "<string>", Variable(x), Variable(x), ...

where: <string> = text string with embedded time, date, and variable descriptors

See the LCD command in this section for complete details of the descriptors and how they are used.

Example:

```
DEFINE Reading = Variable(0)

BEGIN
IF Input(4)=ON THEN
  Reading = (ADC(0) * 50) / 256
  Network = “TERM3 S=Value = %P0\n”, Reading
END
```

OFFHOOK Take phone line off hook (pick up phone)

The phone line is taken off hook, and the XPRESS program is suspended for 2 seconds to enforce a billing delay. Once control passes back to the XPRESS program, further actions may be performed immediately.

Only works in the sequential section.

Syntax: OffHook

Hardware: HCS-DTMF or SpectraSense

Example:

```
!
! Answer the phone on the third ring,
!  give a greeting message, and hang
!  up again
!
IF Rings>=3 THEN
  OffHook
  SayW"This is an automated greeting."
  OnHook
END
```

Put phone line on hook (hang up phone) — ONHOOK

The phone line is put on hook. Control passes back to the XPRESS program immediately.

Syntax: `OnHook`

Hardware: HCS-DTMF or SpectraSense

Example:

```
!
! Answer the phone on the third ring,
!  give a greeting message, and hang
!  up again
!
IF Rings>=3 THEN
  OffHook
  SayW"This is an automated greeting."
  OnHook
END
```

Test or set a local digital output — OUTPUT

All local digital output numbers are predefined and depend on the hardware being used. See Appendix B for a table of supported hardware and corresponding output numbers.

Syntax: `Output(n) = ON/OFF`

where: n = output number (0–207)

Example:

```
IF Input(3)=OFF AND Output(8)=ON THEN
  Output(8)=OFF
END
```

RANDOM Choose a random number

A pseudorandom number within a given range is selected and may be used in either comparisons or assignments. The number chosen is always between zero and an upper limit passed to the function.

Syntax: Random(n)

where: n = upper limit for random number (1–255)

Examples:

```
!
! Give the house a lived-in look by
!  turning on a light a random amount
!  of time (but no more than 20
!  minutes) after a certain time of day
!
DEFINE Light = Module(L4)
BEGIN
IF Time=18:00:DY THEN
  Timer(70) = ON
  Variable(0) = Random(20)
END
IF Timer(70)>Variable(0) THEN
  Light = ON
  Timer(70) = OFF
END

! To select a random number between
!  20 and 80, use the following:
!
Variable(2) = 20 + Random(60)
```

Set PL-Link refresh period REFRESH

The PL-Link will periodically send out ON or OFF commands to all X-10 modules that have received ON or OFF commands since the PL-Link's last reset. The Refresh command sets the period in which all modules are reset. The system defaults to no refresh. Refresh is not supported when using the SpectraSense's on-board X-10 hardware.

Syntax: Refresh = r

where: r = refresh period in minutes (0–99)
(set to 0 to turn off refresh)

Hardware: PL-Link

Example:

```
IF Input(23)=ON THEN
  Refresh=10
END
```

All X-10 modules that have been referenced by the PL-Link since its last reset are refreshed within a 10-minute period. The frequency at which commands are sent to the power line depends on how many modules must be refreshed. For example, if the refresh period is set for 10 minutes and there are 20 modules in use, a new command is sent every 30 seconds. After 10 minutes, all modules will have been refreshed and the cycle starts again.

True after reset RESET

On a system reset or when a new XPRESS program is loaded, this condition is true for the first pass through the program and is always false thereafter. It is used to set up initial or default system conditions.

Syntax: Reset

Example:

```
IF Reset THEN
  Refresh = 10
  Module(D2) = ON
  Variable(4) = 0
END
```

RESETIO Reinitialize all 8255 I/O ports

To make the system more fail-safe in noisy or harsh environments, it is sometimes desireable to be able to periodically reinitialize all the hardware 8255 I/O ports to ensure they are properly configured for inputs and outputs. ResetIO reinitializes all system 8255s to their proper configurations. This command has no effect on the SpectraSense's on-board I/O ports.

One note of caution: Whenever an 8255 is reconfigured, it clears all its output ports. The HCS automatically updates the output ports, but there may be a period of a few tens of microseconds when the outputs aren't at the values you set.

Syntax: ResetIO

Example:

```
!
! Reinitialize the direct I/O ports
!   every hour
!
IF Reset THEN
  Timer(80) = ON
END
IF Timer(80)>=60 THEN
  ResetIO
  Timer(80) = ON
END
```

RINGS Returns the number of rings detected currently

Rings is used to determine how many consecutive rings have been detected on the current incoming call. A standard telephone ring signal lasts six seconds (two seconds of ring, four seconds of silence). The Rings parameter is incremented each time a ring is detected and is cleared to zero when more than six seconds has elapsed since the last detected ring signal. It returns a value in the range of 0–255.

Syntax: Rings

Hardware: HCS-DTMF or SpectraSense

Example:
```
!
! Answer the phone on the third ring,
!  give a greeting message, and hang up
!  again
!
IF Rings>=3 THEN
  OffHook
  SayW"This is an automated greeting."
  OnHook
END
```

Speak a text string — SAY

Using the HCS-Voice text-to-speech synthesizer, virtually anything may be said by the system, including variable values. All commands supported by the text and phoneme modes of the HCS-Voice may be used. The normal command character is a nonprintable character, so a tilde character (~) is used in front of any commands. (See the HCS-Voice manual for a description of its commands.)

In addition to text strings, variable values may also be spoken. The format of the command is identical to that of the LCD and LPT commands.

Since the HCS-Voice has its own processor, text strings are sent to it very quickly and XPRESS processing proceeds while the text is being spoken. To suspend sequential XPRESS processing until the synthesizer is finished, use the SayW command.

Phrases being spoken by the HCS-Voice may be interrupted by sending it a vertical bar character (|). For example, it’s possible to send a phrase to the synthesizer and wait for a DTMF digit while the phrase is being spoken. If the user presses a button before the phrase completes, a vertical bar preceding the next text string causes the first phrase to be interrupted and the second phrase to start.

Syntax: Say“<string>”, Variable(x), Variable(y), ...

where: <string> = text string with embedded time, date, and
variable descriptors

See the LCD command in this section for complete details of the descriptors and how they are used.

Hardware: HCS-Voice

Example:
```
!
! Assuming the outside light level is at
!  0 volts for darkness and 5 volts for
!  sunlight, speak the light level
!  percentage. Just before speaking the
!  phrase, select a new pitch level.
!
DEFINE Button   = Input(5)
DEFINE LightLevel = ADC(0)
DEFINE Light   = Variable(11)

IF Button=ON AND Button=EDGE THEN
  Light = (LightLevel * 100) / 26
  Say"~60P The outside light"
  Say"is at %P1 percent.",Light
END
```

SAYW Speak a text string and wait for it to finish

SayW is identical to Say, but it suspends sequential XPRESS processing until the text-to-speech synthesizer has finished speaking the phrase. One example of where this is useful is when a phrase is to be spoken just before hanging up the phone. If Say is used, the phone is hung up right after the start of the phrase. With SayW, the phrase completes before the phone is hung up.

Only works in the sequential section.

See the description of Say for more details.

SECOND Current second

Retrieve the current second. Second is in the range of 0–59.

Syntax: Second

Examples:
```
IF Input(8)=ON THEN
  Variable(5) = Hour
  Variable(6) = Minute
  Variable(7) = Second
END
```

Test current time of day — TIME

Compare the current time to a given time of day. Hour, minute, and day of week are tested.

Note that testing is based on a 24-hour day that lasts from midnight to midnight.

Syntax: Time <op> hh:mm:dd

where:
```
<op> = "=", "<>", ">", "<", ">=", "<="
hh = hours (00–23)
mm = minutes (00–59)
dd = day of week (MO,TU,WE,TH,FR,SA,SU,DY,WK,EN)
               DY = daily, WK = weekday, EN = weekend
```

Examples:
```
IF Time=08:00:DY THEN
  Output(7)=OFF
END
```

At 8:00 in the morning every day, turn output number 7 off.

```
IFA Time>23:00:MO OR Time<6:00:TU THEN
  IF Input(9)=ON THEN
    Module(E5) = ON
  END
  IF Input(9)=OFF THEN
    Module(E5) = OFF
  END
END
```

Turn on the porch light (controlled by X-10 module E5) when motion is detected on input number 9 and off again when the motion stops. But do this only if it is between late Monday night and early Tuesday morning.

Note the use of the OR operator in the expression. The time of day and day of week are tested separately. The first expression is only true on Monday and the second is only true on Tuesday. Using an AND operator, as you might be inclined to do at first glance, results in the expression always being false. Changing the day of week to daily—

```
IFA Time>23:00:DY OR Time<6:00:DY THEN ...
```

—doesn't change the need to use an OR. The first expression is only true when Time is greater than 23:00 and less than 23:59. The second expression is only true when Time is greater than 0:00 and less than 6:00. As before, using an AND results in the expression always being false.

When testing times that don't straddle two days, an AND must be used.

```
IFA Time>6:00:DY AND Time<23:00:DY THEN ...
```

In this case, if the OR operator were used, the expression would always evaluate true. The first expression is true if Time is greater than 6:00 and less than 23:59. The second expression is true if Time is greater than 0:00 and less than 23:00. Combining these with an OR always results in a true state. Combining them with an AND gives the desired result.

TIMER Retrieve the current state of a timer

When a timer is tested for ON or OFF, the result of the test reflects the state of the timer (e.g., if a timer is tested for ON, the test reults in TRUE is the timer is indeed on). Timers may also be tested against numerous system conditions or used in equations or assignments.

Timer numbers 0–63 are incremented every second while timers 64–127 are incremented once a minute.

Syntax: Timer(n) <op> <val>

where: n = timer number (0–127)

Examples:

```
IF Timer(3)>=8 THEN
  Module(G7) = OFF
END

IF Input(7)=ON THEN
  Variable(5) = Timer(8) * 5
END
```

There are actually two sets of timers defined: one that counts seconds and another that counts minutes. Timers 0–63 are incremented once per second while timers 64–127 are incremented once per minute. Minute timers may trigger any time during the minute preceding the setpoint (i.e., a timer tested for 5 minutes may show true any time between 4 minutes 1 second and 5 minutes).

When off, all value comparisons result in false. When turned on, timers are cleared to 0 and begin counting. To clear a timer that is already running, it is not necessary to stop the timer. It may simply be restarted (e.g., if a timer is started, runs for 15 seconds, and is then turned on again, it is cleared to 0 and continues running from there).

Start or stop a system timer

TIMER

Start or stop a system timer. When a timer that is already on is set to ON, it is simply cleared to zero and allowed to continue running.

Syntax: Timer(n) = ON/OFF

where: n = timer number (0–63 for seconds, 64–127 for minutes)

Example:
```
IF Input(4)=EDGE THEN
  Timer(7) = ON
END
```

Test or set a variable

VARIABLE

Variables may contain any 16-bit number and may be tested against or set to many different system conditions. Variables are considered off or false when set to zero and on or true when set to any nonzero value.

Syntax: Variable(n)

where: n = variable number (0–127)

Example:
```
IF Variable(2)>5 THEN
  Output(4) = OFF
  Variable(2) = 0
END
```

Suspend sequential XPRESS processing for a fixed time

WAIT

Sometimes it's helpful to insert pauses that actually suspend processing (timers may be used to insert delays in the continuous section that don't suspend processing). Wait may be given a constant value that generates a pause from 0.1 to 25.5 seconds.

Only works in the sequential section.

Syntax: Wait(n)

where: n = suspend time in tenths of a second (1–255)

Example:

```
!
! Generate a half-second pulse on
!  a direct output
!
IF Time=7:00:DY THEN
  Output(10) = ON
  Wait(5)
  Output(10) = OFF
END
```

YEAR Current year

Retrieve the current year. Year is in the range 0–99.

Syntax: Year

Example:

```
IF Year>=90 THEN
  LCD(0) = "This the '90s\n"
END
```