XPRESS

Reference Manual

for the HCS II

Release 3.62
3/10/98

Creative Control Concepts
4479 Mebane Rogers Road
Mebane, NC 27302-9732
http://www.cc-concepts.com/

Table of Contents

1. System Overview

Welcome to the world of home and building automation. An automated building is a foreign concept to many people used to living in a world of manual control. The idea of lights turning on and off by themselves, drapes automatically opening or closing, or chimes ringing in response to a car in the driveway may take some getting used to. However, once you've lived with the conveniences for even a short time, you quickly come to rely on them.

The Creative Control Concepts Home Control System II is a new-generation control system with roots in industrial control. With the XPRESS (eXpandable, Programmable, Real-time Event Supervisory System) programming language, the HCS-II gives users unprecedented control over their environment.

A basic HCS-II setup consists simply of a Supervisory Controller (SC). The base HCS-II SC gives the user 16 digital inputs, 8 digital outputs, and 8 channels of 8-bit analog-to-digital conversion. Optional plug-in expansion boards increase the direct digital I/O to 64 inputs and 64 outputs. Other plug-in expansion boards include the HCS-PLIX which provides a 2-Way X10 interface, the HCS-DTMF which provides telephone/DTMF control, and the HCS-Voice which provides true text to speech voice capability

By adding one or more RS-485 network modules, the system's reach increases by leaps and bounds. Up to 31 network modules may be connected to the system using inexpensive twisted-pair wiring. A single twisted pair (up to 4000 feet long) run throughout the house is all that's needed to tie the system together.

Numerous types of RS-485 network modules exist. The MCIR-Link allows you to issue commands to the HCS-II using a standard trainable hand-held IR remote control. It also enables the HCS-II to send infrared commands to virtually any piece of consumer electronics equipment. The LCD-Link/PIC-LCD gives the user a gateway to the HCS-II with various size LCD displays. Each DIO-Link/PIC-DIO has 8 bits of digital I/O and the PIC-DIO allows you to connect a printer to your HCS-II system. The AMAN-Link provides a very versital interface which can be configured for 8 bit of digital I/O, 4 8-bit Analog to Digital inputs, 2 12-bit Analog to Digital inputs, Pulse Width Modulation, Pulse Totalizer, Dallas iButton reading, and more. The PIC-TV module allows you to interface your television to your HCS-II and output over 300 characters of text and graphics. You can even overlay the text on an existing video signal. There is even a tiny Mini-Link which provides 4 digital I/O ports, 3 of which can be configured as 8-bit ADC inputs and it all fits on a tiny 1.5" square circuit board. Up to 8 of each kind of module may be connected to the network (up to the system-wide limit of 31 modules total).

An IBM PC or compatible computer is used for initial setup and programming of the HCS-II. When all is working as expected, the PC may be removed from the system until the next time a change is required. However, the included HOST program will provide you with a wealth of information about your system when your PC is connected to your HCS-II.

1.1 Definition of Terms

Here's a list of terms used throughout this manual you should understand before you proceed.

Supervisory Controller (SC)—the main board that controls the whole HCS II system.

Network Module —a generic term used to describe any of the HCS II network modules.

Local Input (or Direct Input)—any digital input connected directly to the SC or Buffer Terminator (BUF-Term) board (which conditions those signals).

Local Output (or Direct Output)—any digital output connected directly to the SC or Buffer Terminator (BUF-Term) board.

Netbit—any digital input or output connected to any COMM-Link network module.

Edge—an on-to-off or an off-to-on transition on any digital input.

RS-485—an interface standard used as the basis for the HCS-II network. It defines the use of a single twisted-pair cable to carry all system communications.

XPRESS—eXpandable, Programmable, Real-time Event Supervisory System; the programming language used by the HCS-II.

PL-Link -- The original Power Line/X10 interface for the HCS-II. It has been replaced by the HCS-PLIX. When ever you see PL-Link, HCS-PLIX applies there as well. Note the HCS-PLIX is not a network module, but rather an HCS-II SC daughterboard.

2. Getting Started

If you're like most people, you don't have the patience to read the entire manual before setting up the system to see it work. To satisfy your impatience, we've provided quick setup instructions so you can get your system working in just a few minutes. When you're ready to install the system for real, read the remaining chapters before proceeding

Just a note of warning: *Do not* make any connections other than the ones we suggest below until after you've read the rest of the manual. Failure to heed this warning may result in damage to one or more HCS-II components or to your own equipment. Such damage is *not* covered under any kind of warranty.

2.1 HCS II Quick Setup

The minimal setup we assume here is an HCS180 or HCS2-DX Supervisory Controller (SC), a RBUF-Term relay buffer board, and a PL-Link or HCS-PLIX module. You should also have a push button or a sensor with a normally open output.

Verify that all the jumpers are installed on the SC in their factory default positions as described in the SC hardware manual. Likewise, make sure the jumpers are set up on the PL-Link as shown in its manual.

Connect a 26-pin ribbon cable between J26 on the RBUF-Term board and J4 on the SC. Pay close attention to the location of pin 1 when plugging the cable into each board.

Connect the sensor or push button to the screw terminals under LED1 labelled Vin and GND on the RBUF-Term board.

Make sure your power supply is off and connect it to the power terminals on the RBUF-Term board (T1).

Similarly, connect the 9–18-V power supply to the power terminals on the PL-Link (T2). Again, they are labeled V+ and GND. The HCS-PLIX draws power directly from the SC.

Connect the 9–18-V power supply to the power terminals on the HCS2-DX board (J12). Once again, they are labeled V+ and GND.

Connect a pair of wires between T1 on the PL-Link and pins 1 and 2 of J1 on the SC. Pin 1 should go to the "+" on the PL-Link and pin 2 should go to the "–". Making this connection backwards won't damage anything, but the network won't work right.

Plug one end of the modular telephone cable into the phone jack on the PL-Link/HCS-PLIX.

Plug the other end of the same telephone cable into the TW523 and plug the TW523 into a wall outlet.

Set up an X-10 lamp module for housecode L, unit 1. Plug the module into an outlet and plug a lamp into the module. Similarly, set up another lamp module for L2 and plug it in with a second lamp. Be sure the power switches on the lamps are on.

Connect a serial cable between J2 on the SC and COM1 on a PC-compatible. Be sure pin 1 on the serial cable matches the pin 1 location on the connector on the board.

Double check ALL power connections to ensure nothing is wired backwards (especially the SC power connections).

Finally, turn the power supply on. The LED on the PL-Link should blink slowly. The LED on the SC remains off.

2.2 Quick Program

Type the sample program shown below into a plain text file on your PC using a text editor or a word processor in text-only mode. In some word processors, you must make a special effort to request that the program save your file as plain ASCII text. Call the file EVENTS.HCS. Be sure your word processor does *not* add any extra control characters or formatting commands.

```
!
! Sample HCS-II control program
!
CONFIG SC = HCS180
CONFIG PL-Link = 1

Display Modules = L

DEFINE Sensor    = Input(0)
DEFINE Lamp1     = Module(L1)
DEFINE Lamp2     = Module(L2)
DEFINE LampDelay = Timer(1)

BEGIN

IF Sensor=ON THEN
  Lamp1 = ON
END

IF Sensor=OFF THEN
  Lamp1 = OFF
END

IF Sensor=EDGE AND Sensor=ON THEN
  Lamp2 = ON
  LampDelay = ON
END
```

```
IF LampDelay>=5 THEN
  Lamp2 = OFF
  LampDelay = OFF
END
```

The program simply watches the input and turns the first lamp on when the input goes on, then off when the input goes off. It also turns the second lamp on when the input goes on, but turns the lamp off five seconds later regardless of the state of the input.

When you're done creating the file, exit the text editor and type COMPILE. The XPRESS compiler checks your program for proper syntax and creates a binary equivalent called EVENTS.BIN. If COMPILE finds any typing errors, fix them and compile the program again.

After a successful compile, type HOST to enter the host program. You should get a screen full of windows that display various system status information. Press the "C" key followed by "T" to set the SC's time and date. Then, press "F" followed by "L" to load your compiled program into the SC. When HOST tells you the load is successful, your program is running and should respond to your sensor or push button as described above.

If HOST tells you that the SC isn't responding to either command, there is probably a problem with the serial connection between your PC and the SC. Double check all connections.

The PL-Link must also be active on an HCS II system for any X-10 commands to be sent to the power line. Make sure the Network Mods window shows a "*" next to the PL-Link label. If either "–" or "E" show up, double check your network connections.

Your HCS-II is now operational. Feel free to experiment by modifying the sample program for other behaviors. Be sure to recompile the program each time you make a change. When you're ready to begin the actual installation of your system, read the rest of this manual before making any more connections.

3. System Design

The HCS-II provides you with an incredible amount of flexibility in how you set up your system. We recommend you take some time to consciously design your overall system rather than randomly run wires and make connections. Take a few moments to think about how you want your system to operate.

> Will simple X-10 power-line control meet all your control needs, or will you need to run some wire?

X-10 signals travel over existing power lines, so additional wire isn't necessary. While the X-10 system is perfect for turning lights on and off, it can be somewhat unreliable and slow for other types of communication such as sending sensor status back to the SC.

> If you want to run wire for motion detectors, magnetic switches, temperature sensors, and so forth, where are those sensors going to be located?

It's always a good rule of thumb to run twice as much wire as you think you'll need. Eight-conductor (four twisted pair) cable is a good choice if you're pulling wire anyway. You may also be able to run a group of sensors back to a single DIO-Link or ADIO-Link, requiring just one twisted pair to be run back to the SC. For example, if you have three bedrooms grouped at one end of the house, you could run all the sensors from those three rooms back to a single DIO-Link located centrally to the rooms.

> How fast a response do you need to a sensor input?

Responses to sensors connected directly to the SC are almost instantaneous when controlling an output connected directly to the SC. Inputs and outputs traveling across the network slow the response down a bit. For example, in a typical setup, sending an X-10 command in response to an input on the SC may take half a second, but turning on an output on a DIO-Link in response to an input connected to a DIO-Link may take up to two or three seconds.

Sensors that require an immediate response should have wires run back to the SC for direct (or local) connection. An example of such a sensor might be a motion sensor at the top of a stairway. How useful is a light that doesn't come on until you're halfway down the stairs? On the other hand, a 1- or 2-second delay won't make any difference if you're sounding a chime in response to a car in the driveway.

The SC should be placed in a central location close to an AC outlet (ideally

3.1 Locating the Supervisory Controller

near the fuse box or breaker panel). Easy access to a telephone line may also be necessary for future Network modules.

Many people prefer to mount a sheet of plywood on the wall in the basement or garage where the SC is to be installed. The SC may be mounted on the wood at chest level to make working on it easier. Additional expansion boards that are to be located nearby (such as a PL-Link) may also be mounted right on the wood. Buffer boards and connector strips are easy to mount and make running wires from place to place clean and easy.

Alternatively, you might mount the SC and any buffers and connectors inside a standard metal alarm-type box with a door on it. The SpectraSense 2000 already comes packaged in such an enclosure.

Don't forget you'll need to connect a PC to the SC for programming, so there should be either a PC nearby, a portable PC (such as a laptop) available, or a serial cable run from the SC to your PC. Keep in mind that if you run cable for an RS-232 connection, the cable should be less than 50 feet long and kept away from noise sources like blower motors, oil burner transformers, and power lines in general.

3.2 Power Supplies

The original HCS180 requires +5 V while the newer HCS2-DX and the other Network modules all accept any DC voltage between 9 and 18 V. A single power supply may be used to power the whole system, or separate power supplies for each Network may be used. If you're using a BUF-Term parallel buffer board with your SC, you may run a 9–18-V power supply into it and it will provide regulated 5 V for the SC, allowing the use of a single 12-V (or similar) power supply for the whole setup. The SpectraSense comes with its own power supply.

If you elect to use a single power supply for the system, you might run the power in the same cable as the main network to make powering network modules easy. In such a setup, you'd have to run at least four-conductor (two twisted pair) cable to each node location. Be sure to allow about 200 mA at 12 V for the SC and RBUF-Term combination, plus an additional 200 mA for each Network module on the network.

For an example setup consisting of an SC, RBUF-Term, and three Networks, your power supply should be capable of supplying at least 800 mA at 12 V. In such a case, it would be prudent to make sure the supply is good for at least 1 or 1.5 A to allow for future expansion.

The memory on the SC is battery backed, so in the event of a power failure, your XPRESS program won't be lost. When power is restored, the SC will continue to execute your XPRESS program as if power had never been lost. Be aware, however, that other devices in the house that aren't battery backed (such as X-10 modules) may need to be manually reset after a power outage.

3.3 Setting Up the Network

In a very basic HCS-II setup where only an SC is used with direct inputs and outputs, a network isn't necessary because there aren't any network modules to connect.

In an only slightly more complicated system where the only network module is a PL-Link, all that's really necessary is a six-inch piece of cable to connect the PL-Link to the SC. No additional cable needs to be run and the wiring to the PL-Link can be virtually anything.

In systems where the network extends beyond six inches, some care must be taken when selecting and running the cable to ensure error-free operation of the network.

3.3.1 Selecting Cable

The most important factor in selecting a cable is to be sure it contains twisted pairs. Most common 4-conductor telephone cable runs the wires in parallel and is not suitable for use in this system. The minimum cable must contain two twisted wires (a single twisted pair). The wires may be solid or stranded, and are typically 22 gauge down to 26 gauge. If you are running power to the modules, then cable with two twisted pairs (four wires) is necessary. If you're installing new cable anyway, we recommend you run cable containing at least four twisted pairs (eight wires) to account for expansion. The emerging CEBus home automation standard also calls for Category 5, four-twisted-pair cable, so you'll be ready for future CEBus devices as well.

3.3.2 Running Cable

The network topology is left up to you to decide. A single cable run may be made throughout the house, where the cable starts at the SC, then goes from one room to the next. This configuration is the easiest to run and requires the least amount of cable. A problem with this approach is the possibility that a wire in the cable could break somewhere in the middle of the run, cutting off all devices downstream of the break from the rest of the system.

The alternative we recommend is to run separate cables from the SC to each room in the house. Some larger rooms may require multiple cable runs. While this approach takes more time in the beginning and requires more cable, it offers a good deal more flexibility, and troubleshooting is easier when you can isolate individual rooms. As above, the CEBus specification calls for separate cable runs from a central location to each room, so you're once again preparing for the future in more ways than one.

You may also find it easier to connect sensors directly to the SC if you run separate cables to each room. For example, if you run four-twisted-pair cable to each room, you can use two of the pairs to carry the network and power to, say, a DIO-Link that has window and door sensors tied to it, and the other two pairs to carry contact closures from a pair of motion detectors back to the SC for direct connection.

When installing the network cabling, be sure the cable is at least six inches away from any AC power cable. *Never* install network cable in the same conduit as AC power-line cable. If the network cable must cross the path of an AC cable, be sure the two paths cross at right angles.

3.4 Analog and Digital I/O

The digital I/O ports on the HCS180, HCS2-DX, LCD-Link, DIO-Link, and AMAN-Link are low-current, TTL-compatible ports. In almost all installations, additional buffers and drivers are necessary when making connections to these ports. Under no circumstances should the ports be used (when configured as outputs) to drive lamps, relays, bells, or other devices directly. Sensors may be connected directly to the ports when used as inputs, but extreme care must be taken to ensure the voltage on the inputs doesn't drop below 0 V or go above 5 V.

To protect your HCS II hardware, we strongly advise the use of our BUF-Term parallel buffer board. Each input buffer on the board withstands voltages from –30 V up to +30 V and each output driver handles 500 mA at 50 V (with 20% duty cycle) or 175 mA at 50 V (with 100% duty cycle). The digital inputs and outputs on the SpectraSense already use such buffers and drivers.

Similar precautions must be taken when dealing with the analog I/O on the SC and AMAN-Link. The analog inputs must be constrained to a range of 0–5 V, while the outputs must be buffered before they can be used to drive any device. Because the requirements of an analog interface usually differ for every installation, we currently don't have any off-the-shelf analog buffer boards. Be sure to contact Circuit Cellar technical support if you're at all unsure about how to make analog connections in your application.

Damage to boards caused by unbuffered inputs or outputs is not covered under warranty!

Refer to the individual hardware manuals for the SC, RBUF-Term, and Networks for details on connector pinouts and interface limitations. Appendix B lists the ports on each of the boards; whether they are assigned to be inputs, outputs, or both; and how they are referenced from within an XPRESS program.

3.4.1 Signal Polarity

In the XPRESS language, an unbuffered digital input or output at 0 V is considered OFF while an input or output at 5 V is considered ON. Buffers and interface circuitry may ultimately invert inputs or outputs so the ON and OFF labels end up reversed. For example, a simple transistor driver on an output driving a light may work by turning the light on when the output goes to 0 V and off when the output goes to 5 V. In such a case, the XPRESS statement “Output(x) = ON” actually turns the light off.

Similarly, a normally open contact closure might have one contact connected to ground while the other contact is connected to a system input with a pull-up resistor. When the sensor is inactive (or open), the input reads ON. When the sensor closes and grounds the input, the input goes OFF.

You must take all such polarity reversals into account when writing your program. To make a program easier to read, you might use the following definitions:

```
DEFINE OnN = ON      ! Normal "on" state
DEFINE OffN = OFF    ! Normal "off" state
DEFINE OnI = OFF     ! Inverted "on" state
DEFINE OffI = ON     ! Inverted "off" state
```

For all inputs and outputs that use normal (or noninverted) states, you would use the “OnN” and “OffN” pair in your program. For inputs and outputs that get inverted by buffers, use the “OnI” and “OffI” pair.

3.4.2 Response Time

Response time to inputs (analog and digital) depends a great deal on whether the input goes over the network and, if so, how complicated the network is. Sensors that need immediate attention should be connected to the SC. Response time to such inputs can be almost instantaneous. When inputs must go over the network, response times range from about half a second for a simple network to several seconds for a more complicated network.

Analog and digital output response time behaves similarly. Outputs connected to the SC change immediately in response to a XPRESS statement evaluating true, while outputs connected to the network require from half a second to several seconds to be changed depending on network complexity.

3.5 X-10 Power-Line Communications

The X-10 system communicates without additional wiring by sending messages through the home's power lines. The world of AC power was never designed for data and is a harsh, difficult environment in which to send packets of information. While the vast majority of X-10 installations work flawlessly, any number of factors in the home can combine to make such communication unreliable in some cases.

3.5.1 Signal Bridging

If you find you can control some X-10 modules reliably, but can't reach others, you're probably having trouble communicating between 110-V legs.

Power enters the house through three wires: two "hot" wires and one neutral wire. The voltage between the two hot wires is 220 V, while the voltage between either hot wire and the neutral wire is 110 V. Appliances such as electric stoves and clothes dryers that use 220 V are connected to both hot wires. The rest of the house is usually wired so half the outlets are connected to one hot wire and the other half are connected to the second hot wire.

X-10 transmissions that originate on one hot wire (or "leg") usually must exit the house, go all the way to the pole on the street, then back to the house to get to the other leg. Most of the time the data packet won't survive such a trip and is lost. The electric stove and clothes dryer mentioned above can often provide a shortcut between legs so the packet can get across, but the shorcut is only there when the appliance is on.

Leviton Manufacturing makes a signal bridge module (LV6201) which can be installed in the fuse box or breaker panel, that provides a continuous path for packets to travel from one leg to the other. Contact Creative Control Concepts for information on how to obtain this module. We recommend only a qualified electrician install this device.

3.5.2 Multiple Transmitters

If you find that commands sometimes don't get through when you're using an X-10 manual control unit, your commands may be colliding with those being sent by the HCS-II. When a group of people are talking and two speak at the same time, understanding what either said is usually impossible. Similarly, when the HCS-II is transmitting and you try to transmit at the same time, both messages are usually lost.

Unfortunately there isn't much that can be done (short of you never using a manual X-10 controller). In most installations, the HCS-II is seldom transmitting, so the chances of a collision are low. If you leave X-10 refresh turned on (see the Refresh command in Chapter 7), your chances of a collision are much greater. In general, if you leave refresh off when you're home, and only turn it on when you're away, collisions shouldn't be a problem.

3.6 Initial Start up

When all the HCS-II hardware is in place, connect the RS-232 cable from the SC to COM1 or COM2 of the host PC. Type the following program into a file using a text editor (as described in Section 4.1) and save it as EVENTS.HCS. Modify the Config statements to reflect the proper number of each kind of Network module you have connected (e.g., if you don't have an LCD-Link in your system, change the "1" to a "0" in the LCD-Link configuration statement).

```
CONFIG SC = HCS180
CONFIG PL-Link = 1
CONFIG IR-Link = 1
CONFIG LCD-Link = 1
CONFIG DIO-Link = 1
CONFIG ADIO-Link = 1

Display Modules = A,B,C

BEGIN

IF Reset THEN
  Refresh = 0
END
```

Compile the above program using COMPILE as described in Section 4.1.

Run the HOST program as described in Section 4.2 and press the "F" key followed by "L" to load your XPRESS program. If HOST responds with a "sad" tone and a message indicating a timeout, press "F" and "L" again. If the program still won't load, double check the serial connection between the SC and the PC. When the program transfer completes with a "happy" tone, you should immediately see the current status of any network modules, X-10 modules, direct inputs, and direct outputs you've defined in your XPRESS program.

All network modules that are powered up and properly connected to the network should show a "*" in the network status window indicating the module is active. A "–" indicates the module isn't responding at all. It may not be powered up properly or may not be connected to the network correctly. An "E" indicates errors are occurring during network communication. Check all network wiring to be sure the cable doesn't run near any sources of electrical noise (including AC power lines) and check network termination on all Networks. Also, make sure the network connections on the module aren't reversed.

If the X-10 status window doesn't list housecodes A, B, and C, then there is a problem with the PL-Link connections. You should also see either a "–" or a "!" in the network status window next to the PL-Link label. Check the PL-Link's connections as described in Section 2.1.

Now, check any direct inputs connections. The current status of each input is updated once per second in the input window. For example, if you have a motion detector in the room where you're working connected to one of the direct inputs, you should see the input state change whenever you move.

You won't be able to check any output connections until you've written a program to control those outputs.

When the system seems to be working correctly, you may begin writing your first custom XPRESS program. Refer to Chapters 6 and 7 for details about XPRESS.

4. Software on the PC

A PC-compatible computer is used to program and monitor the HCS-II. Once the system is working, the PC may be disconnected or shut off until changes are needed. While the PC software works on most any PC compatible you may have lying around, we recommend at least an 8-MHz PC or an AT-class machine. The software works with a single floppy disk drive and either color or monochrome display.

There are two programs used on the PC: COMPILE and HOST. COMPILE is used to compile XPRESS programs from their original source form to a binary representation used by the SC. HOST is used to set the time, send XPRESS programs to the SC, display system status, clear the system log memory, and transfer logged data from the SC to a disk file on the PC.

4.1 COMPILE

XPRESS programs may be written using any text editor capable of producing flat ASCII (or text) files. If you're using a fancy word processing program (such as Word Perfect, Microsoft Word, WordStar, Volkswriter, or any of the hundreds of others), be sure you save your code as a plain text file that contains no control characters or formatting commands. Make absolutely certain there is a carriage return after the last line of your XPRESS program. It is a good idea to leave a few blank lines at the end of the code to be sure.

Before your XPRESS program can be sent to the SC, the program must be compiled. The compiling process checks your program for correct syntax and value ranges and creates a binary file containing the raw code for the SC.

The form of the COMPILE command line is as follows (items in braces are optional):

```
COMPILE  {filename}
```

where: `filename` is optionally the name of your source file

If the filename isn't given, COMPILE assumes the source file is called EVENTS.HCS. If the extension isn't given, COMPILE assumes an extension of .HCS. The following are valid command lines:

`COMPILE`	compiles the code found in EVENTS.HCS
`COMPILE  MYPROG`	compiles the code found in MYPROG.HCS
`COMPILE  PROG2.TXT`	compiles the code found in PROG2.TXT

COMPILE always creates a binary file called EVENTS.BIN. The HOST program expects your compiled XPRESS program to be in EVENTS.BIN, so be sure not to change the name of the file.

When errors are found during compilation, COMPILE will tell you what kind of error was found and what line it was on. It then displays the offending piece of code.

COMPILE alerts you to several kinds of errors. Syntax errors are usually typographical errors: a misspelled command or a missing statement. A single syntax error may result in more than one error message because COMPILE continues processing your program even after the error occurs. Often fixing a single error will eliminate multiple error messages.

Value errors are often more subtle. They occur when a value you've specified in a comparison or an assignment is too large or too small. For example, if you accidentally specify a time of 28:30 instead of 18:30, COMPILE flags a value error because the hours may only be between 0 and 23.

COMPILE also checks for unbalanced parentheses and IF/END pairs.

4.2 HOST

The HOST program is used to send compiled XPRESS programs to the SC and to display overall system status. HOST displays everything in a text-based windowing environment and works equally well on graphics and text-only systems, and on color and monochrome systems. You should connect the SC to either COM1 or COM2 on your PC before running HOST.

The form of the HOST command line is as follows (items in braces are optional):

```
HOST  {/COM1|/COM2|COM3|COM4}  {/M}  {//}
```

where: `/COM1` tells HOST to use COM1
`/COM2` tells HOST to use COM2
`/COM3` tells HOST to use COM3
`/COM4` tells HOST to use COM4
`/M` forces HOST into monochrome mode (display just black and white; usually used on systems with color adapter cards and monochrome monitors)
`//` displays a short help screen and exits

Examples of valid HOST command lines include:

`HOST`	start HOST in its default mode
`HOST /COM2`	start HOST using COM2
`HOST /M`	start HOST in monochrome mode
`HOST /M /COM2`	start HOST in monochrome mode using COM2
`HOST //`	tell HOST to display a short help screen and exit

The default mode for HOST is to use COM1 and full-color windows.

When HOST is started, it initializes all the windows, puts a command bar at the top of the screen, and displays a window containing information about itself. The information window disappears after a few seconds.

A Microsoft-compatible mouse may optionally be used to select commands and to move, resize, and close any of the on-screen windows. Be sure the correct mouse driver software is installed and the mouse is plugged in before running HOST. In the absence of a mouse, the keyboard may be used to perform all the same functions.

The current active window is highlighted by a bright white frame around it. To select a new window, press the Tab key until the desired window is highlighted. Using a mouse, simply click anywhere inside the desired window to make it active.

The active window may be moved on the screen by using the arrow keys (either the grey keys or those on the keypad). Using a mouse, place the cursor anywhere on the top bar of the window and drag it while holding the left mouse button.

The active window may be resized by using the arrow keys while holding the Shift key. Using a mouse, place the cursor on the lower-right corner of the window and drag it while holding the left mouse button. The correct spot on the lower-left corner is indicated by a single line (as opposed to the double line around the rest of the window).

The active window may be closed by pressing the Del key. Using a mouse, click on the upper-left corner of the window (marked with a star) to close it.

4.3 HOST Commands

There are six main pull-down menus displayed across the top of the HOST screen:

!	program information
File	commands dealing with the PC's disk
Command	commands that may be sent to the SC
Log	commands dealing with data logging
Window	open (or reopen) any closed windows
Modem	commands dealing with an optional modem

To use those pull-down menus, press the key that corresponds to the red letter in the menu name (pressing Alt isn't required). Using a mouse, simply click anywhere on the menu name.

To select a command in the menu (once the menu is displayed), press the key that corresponds to the red letter in the command name. Using a mouse, click anywhere on the command name. It isn't possible to drag the cursor down through the commands. You must perform separate clicks to select the menu, then the command.

To open a window that is closed, select the window name from the Window menu (see below).

The screen setup is saved in a disk file when HOST is exited so the screen retains its look from session to session. To restore the default window sizes and placement, simply delete the HOST.CFG file from the disk.

The following sections further describe the commands in each pull-down menu.

4.3.1 ! Menu

About…	Display information about the HOST program including its version number. The display window is automatically closed after a few seconds.

4.3.2 File Menu

Load	Load a new XPRESS program from disk and send it to the SC. The command expects to find your compiled XPRESS program in a file called EVENTS.BIN. If that file can't be found, an error message is displayed. Errors are also displayed if your XPRESS program was compiled with the wrong version of COMPILE (COMPILE's version must match the version of the EPROM installed in the SC) or if your compiled program is too big to fit in the SC's memory.

Quit	Quit HOST. Your screen is restored to the state it was in when HOST was started.

4.3.3 Command Menu

Set Time	Read the current time and date from the PC's internal clock and set the SC's clock/calendar accordingly. Be sure you set the PC's clock and calendar to the correct time and date before running HOST.
Set Input…	Override a system digital input and force it to either a high (1) or a low (0) value. To reset the input to normal operation, select Transparent. This command is used to test sections of XPRESS code from the HOST screen without having to physically toggle a system input to initiate an action.
Set Output…	Set a system digital output to either a high (1) state or a low (0) state.
Set X-10…	Send an X-10 command to the power line.
Set ADC…	Override a system analog input and force it to a specific value. To reset the analog input to normal operation, select Transparent. Any value in the range of 0–4095 may be set on any analog input channel regardless of the resolution of the ADC normally associated with that channel. When set to Transparent, the channel resolution reverts to reflect the hardware present.
Set DAC…	Set a system analog output. Since the only analog outputs are on ADIO-Links, this command only applies to applications that contain at least one ADIO-Link.
Set Netbit…	Set a system netbit to either a high (1) or a low (0) state.
Voice…	Send a text string to be spoken to the text-to-speech synthesizer board. The HCS-Voice1 board must be installed in the system for this command to work. See the Say command for more details on forming the text string. Press Return alone to exit this command.
Network…	Send a text string to the system RS-485 network. This command should only be used by people who are familiar with constructing commands for network modules. The command should start with the module address, followed by the command. The SC takes care of adding the proper header byte and checksum.

4.3.4 Log Menu

Clear	Clear the log memory. The log memory should always be cleared by using either this command or the ClearLog XPRESS command before any logging takes place. This command also gives you a way to wipe out any old data and start logging new data at a known time.
Size	Displays the number of data entries in log memory. The value is in the range of 0–4095.
Dump	Dump logged data from the SC's memory to a disk file on the PC. The data is always stored in a file called LOGDATA.BIN. If you want to save any old data previously dumped to disk, be sure to rename the old data file before performing a new dump. See Chapter 8 for more information on data logging.

4.3.5 Window Menu

Any screen window that is closed may be opened by selecting it from the Window menu. What information is displayed in each window is indirectly controlled by what hardware is defined at the start of the XPRESS program being run.

Note that the Message window is automatically opened if it is closed and a console message is received from the SC. See the description of the Console command in Chapter 6 for more details.

4.3.6 Modem Menu

Init…	Set a new modem initialization string. The default string used by HOST is AT&FLT&K3&Q5. If your modem requires something different to work properly, you may use this command to define a new init string. The string is saved in the HOST.CFG file upon exit from HOST, so you only have to set the init string once. Be sure to start the command with "AT" as shown above.
Dial…	Instruct the modem to dial a phone number and attempt to connect to a remote HCS-II. Spaces, hyphens, and parentheses in the phone number are ignored. HOST uses tone dialing by default. To use pulse dialing, include a "P" at the start of the phone number. Insert commas anywhere in the phone number to include 2-second pauses. It is normal for some modems to pause briefly after dialing the first digit before continuing with the rest of the number.
Hangup	Instruct the modem to break an existing modem connection and hang up the phone line.

If there is a problem with the connection between the PC and the SC, HOST times out during most of the commands that require a response from the SC. If a timeout occurs, a "sad" beep sounds and a message pops up informing you of the timeout and asking you to try again. If the timeouts persist, check the physical connection between the PC and the SC.

Successful transfers of information during most commands result in a "happy" beep and a message that informs you of the success.

4.4 HOST Windows

HOST displays HCS-II system status information in individual windows on the PC's screen. Each type of information has its own window. The size and position of all windows may be changed using a mouse or the keyboard as described above. The content of each window depends on what hardware was defined at the top of the XPRESS program being run.

The X-10 module status window displays the housecodes called out using the DISPLAY configuration keyword in the XPRESS program. Any modules that are on or dimmed (regardless of dim level) are designated by a "1" on the screen. Modules that are off or not used show up as "0" on the display.

The network-module status window displays the status of only those network modules defined by CONFIG statements in the XPRESS program. The following summarizes the three possible status indicators:

> * : module active and responding normally
> – : module didn't responded when polled by the SC
> E : response from module contained an error and was discarded

An occasional "–" indicates there might be some noise on the network and shouldn't be a concern. Any module showing a consistent "–" should be checked out. Similarly, an occasional "E" indicates some noise on the network and can usually be safely ignored. Consistent errors indicate a problem with the network which will likely cause improper system operation. A leading cause of network errors is improper termination.

Digital inputs and outputs and netbits are displayed in groups of 24 horizontally (though a row may not contain a full 24 bits). A "1" indicates the bit is on and a "0" indicates the bit is off. The starting bit number for each row is shown at the left of each row. The numbers across the top of the rows are offsets from that starting bit number. For example, the first row might have "000" at the left, indicating the leftmost bit in the row is 0, followed by 1, then 2, and so forth. If the next row has "016" at its left, then the bit numbers in that row are 16, 17, 18, and so forth.

Which inputs, output, and netbits are displayed depends on the hardware defined with the CONFIG statement in the XPRESS program. For example, if one ADIO-Link is defined as present in the system, netbits 96–119 are displayed on the HOST screen automatically.

Analog input and outputs are displayed similarly, but with just two values (analog inputs) or four values (analog outputs) per row.

5. Modems and Caller ID

It is sometimes not convenient (or possible) to have a PC close to the HCS-II to check its status, load a new XPRESS program, or dump logged data to disk. It is possible to connect a high-speed modem to the HCS-II and access it using a PC with a modem from anywhere there is a telephone.

When connected by modem, all of HOST's capabilities are available exactly as if the PC and HCS-II were directly connected.

For example, a professional installer may want to offer a service where he updates a customer's XPRESS program upon request (for a fee, of course). Rather than drive to the customer's location to load the updated program into the HCS-II, he can simply call the customer's HCS-II from his office and do it remotely.

Likewise, an HCS-II may be installed in a remote location to perform control functions and log status information. Rather than go all the way out to the site, it's possible to call the HCS-II, dump the logged data to disk for further processing, and then clear the log memory for new entries.

Most name-brand modems that support speeds of 9600 bps (V.32) or faster (V.32bis and V.34) should work just fine.

5.1 HCS/SS Modem Connection

The HCS-II needs an external modem fitting the above description. To connect the modem to the HCS-II, you must use a null-modem cable that swaps pins 2 and 3. A typical 25-pin null-modem cable also swaps pins 4 and 5 and pins 6 and 20. At the very minimum, pins 2 and 3 should be swapped, pins 4 and 5 should be tied together on the PC end, and pins 6, 8, and 20 should be tied together on the PC end.

Two XPRESS commands set up the modem on the HCS-II end. ModemRings tells the modem how many rings should go by before it answers the phone. ModemInit sets up a custom modem initialization string. The default initialization string built into the HCS-II should work fine for most modems, so do *not* use ModemInit unless the built-in string doesn't work with your modem. See Chapter 7 for more details on these two XPRESS commands.

You must load the initial XPRESS program (containing the ModemRings command) into the HCS-II using a PC connected directly to the HCS-II. Once that program has been loaded, simply unplug the PC and plug in the modem (using the null-modem cable). The HCS-II automatically senses the presence of the modem and initializes it accordingly.

To reconnect the PC directly to the HCS-II, simply unplug the modem and plug in the PC (using the straight-through cable). If nothing appears on the HOST screen after reconnecting the PC, press Enter and the display should become active.

5.2 PC Modem Connection

Once the HCS side has been set up, it's time to connect a modem to the PC that will be used to call the remote HCS. Simply connect any modem that matches the description given above to the PC. When using an internal modem, no cabling is necessary. When using an external modem, use a straight-through serial cable to make the connection.

Next, run HOST, making sure to tell it which COM port the modem is connected to (if other than COM1). The status windows open on the screen, but no data is displayed. Select the Modem menu, then the Dial command. You will be prompted for a phone number. Enter the phone number the HCS-II is connected to.

HOST uses tone dialing by default. If you must use pulse dialing, add a “P” to the start of the phone number you enter. Spaces, hyphens, and parentheses in the phone number are ignored. Commas may be used to insert 2-second delays in the dialing sequence.

HOST then calls the number you entered and attempts to connect with the remote HCS-II. If successful, data is displayed on the screen just as if the PC and the HCS-II were connected directly. All of HOST's functions are available for use when connected via modems. When you want to disconnect from the remote HCS-II, select the Modem menu and the Hangup command.

5.3 Caller ID

Certain modems on the market today also support Caller ID. Caller ID is a service provided by your local phone company. When active, a packet of data is sent by the phone company between the first and second rings. This packet includes the date and time of the call plus the full 10-digit phone number of the calling party. Some phone companies also include the name of the calling party.

The HCS-II can receive the Caller ID data from modems that support this feature and pass it along to your XPRESS program for further action. (Note that your local phone company must offer Caller ID service in your area and you must pay them a monthly fee to receive it. Contact your local phone company for more information.) The HCS-II only processes the date, time, and number data. Any name data is ignored and is not available to XPRESS.

At this time, we have tried the Caller ID support of only Supra modems (specifically, the SupraFAXModem 144LC). Other brands do support the feature, but we can't guarantee they will work properly with the HCS-II. In general, any modem that uses the Rockwell modem chipset and whose manual specifically states that it supports Caller ID should work. NOT ALL MODEMS THAT USE THE ROCKWELL CHIPSET ARE CAPABLE OF RECEIVING CALLER ID DATA. If your modem's manual makes no mention of the feature, then your modem almost certainly doesn't support it. The Caller ID feature in Practical Peripherals modems is *not* supported at this time.

To use the HCS-II Caller ID support, connect a modem with Caller ID as described in Section 5.1 and include the command "CallerID = ON" in your XPRESS program. When the phone rings, the Caller ID data is available to your XPRESS program shortly after the first ring. You may then have the HCS-II log the call, display the information, or even speak the name of who is calling. See the Caller ID commands in Chapter 7 for more details.

6. XPRESS Program Structure

The HCS-II is programmed using a language called XPRESS. It is an easy-to-use language similar to BASIC, but tailored to the needs of home and building automation. You needn't be an expert programmer to use XPRESS, but knowledge of rudimentary programming concepts is helpful.

XPRESS programs consist of three sections: configuration, continuous control, and sequential control.

The configuration section defines what processor board is being used as the SC, what hardware is on the SC, how many of each kind of network module is present on the network, and what X-10 housecodes should be displayed by the HOST program. It also contains all the label definitions.

The two control sections make up the bulk of the XPRESS program and consist of a series of IF/THEN/ELSE statements. The IF portion is made up of any number of system conditions combined using Boolean operations. When the IF portion is true, the list of actions in the THEN portion is carried out. When the condition is false, any actions in the optional ELSE portion are executed instead.

The continuous control section contains any code that must always operate in a continuous fashion without pauses or delays. The sequential control section contains code that may pause waiting for user input or some action to complete. Most of your code will go in the continuous section. Program statements that only work in the sequential section (e.g., DialTone, SayW, and Wait) are clearly marked in the next chapter.

The Begin keyword denotes the end of the configuration section and the start of the control sections. The Sequential keyword indicates the end of the continuous control section and the start of the sequential section.

The following is a basic outline of the structure of an XPRESS program:

```
CONFIG …
DISPLAY …                   Configuration section
DEFINE …

BEGIN
    IF|IFA {NOT}<expr1> <op> {NOT}<expr2> ... THEN
      <action1>; <action2>
     …
  ELSE                                 Continuous section
      <action1>; <action2>
     …
  END
```

```
SEQUENTIAL
  IF|IFA {NOT}<expr1> AND|OR {NOT}<expr2> ... THEN
     <action1>; <action2>
     …
  ELSE                              Sequential section
     <action1>; <action2>
     …
  END
```

Throughout the program, case is ignored (i.e., upper- and lower-case letters are treated identically). Tabs and spaces may be used freely. With the exception of text strings and mathematical equations, multiple lines may be used for expressions. Multiple actions may be placed on a single line by separating them with semicolons.

Comments may be placed anywhere in the program by preceding them with an exclamation point. After the exclamation point, the rest of that line is considered a comment.

6.1 IF Expressions

All actions are based on conditions tested in IF statements. An IF statement contains one or more expressions separated by AND or OR and may be preceded by NOT. Evaluation proceeds in strict left-to-right order. Parentheses may be used to prioritize evaluation order.

Expressions are made up of two system values separated by an operator. Valid operators include "=", "<>", ">", "<", ">=", and "<=". Whereever a system value is called for in the next chapter (denoted by <val>), any of the following may be used:

constant (0–32767), ON, TRUE, OFF, FALSE, Variable(n), ADC(n), Timer(n), IRcode(c), Random(n), Hour, Minute, Second, Day, Month, Year, DOW, Rings, CIDnew, CIDmonth, CIDday, CIDhour, CIDminute, CIDarea, CIDexch, CIDnumber

The following may also be used as system values, but only in the sequential section:

DTMFdigit, DTMFnumber, DialTone, CallProgress

Other keywords can only be tested against specific conditions (which are outlined in the next chapter) and include:

Input, Output, Time, Netbit, Module, Reset

Examples of valid IF statements include:

```
IF Timer(4)>=5 THEN ...
IF Hour=5 AND Minute>15 THEN ...
IF ADC(8)>Variable(2) OR (Random(50)<>10 AND
     Input(4)=ON) THEN ...
```

6.2 IF/IFA Differences

Two kinds of IF statements are supported in XPRESS: IF and IFA. When an IF statement is evaluated as true, its THEN portion is executed. On subsequent passes through the XPRESS program, if the condition in that IF statement continues to evaluate true, the THEN portion is skipped (it is executed exactly once for each occurrance of a true condition). The THEN portion is only executed again when the condition evaluates false and then true again.

On the other hand, when the condition in an IFA (IF Always) statement is true, its THEN portion is executed on *every* pass through the XPRESS program as long as the condition remains true.

When a statement contains an ELSE portion, the ELSE is treated according to its companion IF/IFA. That is, when an IF condition evaluates false, the ELSE portion is executed exactly once. On subsequent passes through the program, if the condition remains false, the ELSE portion is skipped. The condition in the IF statement must go true, then false again for the ELSE portion to be executed again.

Likewise, an ELSE associated with an IFA is executed on *every* pass through the program in which the IFA condition evaluates false.

For example, in the following code fragment, the "on" message is sent to the LCD-Link only once when the input goes true, and the "off" message is sent just once when the input goes false. If the THEN and ELSE portions weren't skipped on subsequent passes through the program, the LCD-Link would be overrun with messages.

```
IF Input(5)=ON THEN
  LCD(0) = "Input is on"
ELSE
  LCD(0) = "Input is off"
END
```

In the next example, the LED connected to the output bit is toggled on *every* pass through the XPRESS program. Such a setup is useful to give a visual indication of overall system performance and can be helpful in assessing how much a hardware or software change to the system affects its throughput.

```
IFA Output(2)=OFF THEN
  Output(2) = ON
ELSE
  Output(2) = OFF
END
```

IF/IFA statements may be nested in any way to any number of levels. Care must be taken, however, when nesting IF statements within IFA statements. Take the following example:

```
IFA Input(10)=ON THEN
  Output(2) = ON
  IF Module(D2)=ON THEN
    LCD(0) = "Module is on"
  END
END
```

When input 10 is on, the IFA's THEN portion is executed every time through the program. When module D2 turns on, the command to send a message to the LCD-Link is executed the first time, then skipped on subsequent passes. Now suppose input 10 turns off. The entire IFA THEN portion is skipped because the IFA condition is false. Even if module D2 turns off and then back on, the inside IF is never evaluated so it never sees the module's on-off-on transition. If module D2 is on when input 10 goes on again, the THEN portion associated with the inside IF is skipped because it hasn't seen the on-off-on transition on the module.

It is *very* important to understand how IF and IFA work and what their differences are if you are to write XPRESS programs that work as you expect.

6.3 Actions and Assignments

The THEN and ELSE portions of the XPRESS statement may contain any number of actions and assignments. Items must be on separate lines or be separated by semicolons. IF and IFA statements may also be included in this section. IFs and IFAs may be nested in any combination to any number of levels.

Mathematical equations may be used in assignment statements that deal with numbers. Such equations use signed 16-bit integer math operations and support addition (+), subtraction (–), multiplication (*), and division (/). Evaluation takes place in strict left-to-right order (contrary to traditional math where multiplication and division have a higher precedence than addition and subtraction), however parentheses may be used to alter the order of evaluation.

The negate operator is not supported, and all constants must be positive. An overflow results in a value of 32767 and an underflow results in –32768. No errors are generated in over/underflow situations.

An equation must fit on a single line!

The following examples are valid:

```
Variable(2) = ((ADC(5) * 9) / 5) +32
DAC(3) = 1024 - (Variable(2) * 100)
```

The first example would evaluate the same way with or without parentheses, but their use often makes an equation more readable and eliminates any possibility of confusion. However, the second example would evaluate differently without the parentheses because of the left-to-right evaluation order used by default.

Note that equations using math may *not* be used in the IF portion of XPRESS statements. Math is only available when doing assignments.

6.4 Command Summary

The following may appear anywhere in the XPRESS program:

! <comment>	<comment> = comment ignored by the compiler

The following commands are used in the configuration section:

Config <module> = n	<module> = "PL-Link", "IR-Link", "LCD-Link", "DIO-Link", "ADIO-Link", "DIO+-Link" n = number of modules connected (0–7)
Config SC = <processor>	<processor> = "HCS180", "IND180", "SpectraSense"
Config BUF = b	b = number of BUFIO or BUF50 boards in system (1 or 2; do not count any BUF-Term boards)
Config IND54 = io,io,io,io	io = IN or OUT indicating the direction of each 8255 on the board (there must be four parameters)
Config ADCRES = r	r = ADC resolution (only 8, 10, and 12 are valid)
Config ADCGAIN = g0,...g7	gx = gain for each ADC channel (only 1, 2, 4, and 8 are valid; there must be eight parameters)

Define <label> = <string>	<label> = any alphanumeric word up to 32 characters long with no spaces <string> = any valid XPRESS expression
Display Modules = <list>	<list> = list of housecodes separated by commas

The following commands and system values may be used in both the continuous and the sequential control sections:

ACfailed	AC power has failed since the last test
ACpower	Current status of AC power (ON/OFF)
ADC(n)	n = valid ADC channel (0–135)
AllLightsOn(h)	h = housecode (A–P)
AllUnitsOff(h)	h = housecode (A–P)
CallerID	Turn Caller ID support on or off
CID*xxxx*	Various Caller ID information
ClearLog	Clear logged data from memory
ClearTimers	Stop and clear all system timers
ClearVariables	Clear all system variables to zero
Console="<string>"	<string> = ASCII text string; must be surrounded by quotes
DAC(n)	n = channel number (0–31)
Day	Day of month (1–31)
Dec(Variable(n))	n = variable number (0–127)
DOW	Day of week (1–7)
Hour	Hour of day (0–23)
Inc(Variable(n))	n = variable number (0–127)
Input(n)=ON/OFF/EDGE	n = input number (0–207)
IRcode(c)	c = received IR code (0–255)
LCD(n)="<string>"	n = LCD-Link number (0–7) <string> = ASCII text string; must be surrounded by quotes
Log(i)	i = reference ID of logged data
LogSize	Number of data entries in log memory (0–4095)
LPT(n)="<string>"	n = DIO-Link number (0–7) <string> = ASCII text string; must be surrounded by quotes
MCIR(n)	n = MCIR-Link number (0–7)
Minute	Minute of hour (0–59)
ModemInit	Set a new modem initialization string
ModemRings	Number of rings to wait to answer modem
Module(m)=ON/OFF	m = module (A1–P16)

Module(m)=DIM(n)	m = module (A1–P16)
	n = steps to dim (1–31)

Module(m)=BRIGHT(n)	m = module (A1–P16)
	n = steps to brighten (1–31)

Month	Month (1–12)
Netbit(n)=ON/OFF	n = valid netbit number (0–319)
Netbyte(n)	n = valid netbyte number (0–39)
Network="<string>"	<string> = ASCII text string; must be surrounded by quotes
OnHook	Put phone on hook (hang up)
Output(n)=ON/OFF	n = output number (0–207)
Random(n)	Generate random number (0–n)
Refresh	PL-Link refresh interval (0–99) minutes (0=off)
Reset	True after reset
ResetIO	Reinitialize all 8255 I/O ports
Rings	Number of incoming rings
Say"<string>"	<string> = ASCII text string; must be surrounded by quotes
Second	Second of minute (0–59)
Time <op> hh:mm:dd	hh= hour (0–23); mm = min (0–59); dd = day (SU,MO,TU,WE,TH,FR,SA,DY [daily], WK [weekday], EN [weekend])
Timer(n)	n = timer number (0–127)
Variable(n)	n = variable number (0–127)
Year	Year (0–99)

The following commands may be used only in the sequential control section:

CallProgress(n)	n = number of rings to allow while following progress of outgoing call
DialDigit(Variable(n))	n = variable number containing the digit to be dialed (0–9, 10 [*], 11 [#], 12–15 [A–D])
DialNumber(Variable(n),d)	n = variable number containing the numbers to be dialed (0–9999) d = number of digits to be dialed (1–4)
DialStr("<string>")	<string> = text string containing digits to be dialed (0–9,*,#,A–D, and comma)
DialTone	Check for dial tone
DTMFdigit(t)	t = timeout value in tenths of seconds (1–255; 0.1–25.5 seconds)
DTMFnumber(t)	t = timeout value in tenths of seconds (1–255; 0.1–25.5 seconds)

OffHook	Take phone off hook (pick up)
SayW"<string>"	<string> = ASCII text string; must be surrounded by quotes
Wait(n)	n = suspend time in tenths of a second (1–255)

Some expressions and actions require that specific expansion boards or network modules be installed in the system to work properly. These requirements are noted in the detailed descriptions of each statement.

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 HCSII 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 #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)

```
Example: !
         ! 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

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

Test current time of day

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
```

8. Data Logging

While it is usually pretty easy to set up a control system based on specific events and known environmental conditions, sometimes it's useful to set up one or more sensors, collect data generated by those sensors, and then decide how the control system should work based on the collected data. The collected data may also be used to monitor what happens at the installation while you're away.

Since the Supervisory Controller already has all manner of sensors connected to it, spare memory, a real-time clock/calendar, and some intelligence, it makes a very good data-logging system in addition to a home-control system. Using the Log command (detailed in Chapter 7), single 16-bit values may be stored to memory at any time. A reference ID is attached to the sample to allow data from many different sensors to be logged to the same memory. Separating and sorting of the data is performed at a later time by custom processing software running on the host IBM PC. The current time and date are also saved with the data.

After the SC has been logging data to its memory for a while, you use the Dump command under the Log menu within HOST to dump the data from the SC's memory to a disk file on the PC. The data file is always called LOGDATA.BIN. Be sure to rename an old file before performing a dump if you wish to save the old data.

8.1 Hardware Setup

To perform data logging, an additional 32K static RAM chip must be installed on the Supervisory Controller.

Before doing anything, turn off the power to the SC. Install a standard 62256 (or equivalent) chip in socket U10 on the SC (it is the empty 28-pin socket between the EPROM and existing RAM chip). On the SpectraSense, install the chip in socket U14. Be sure to orient the chip in the same way as the other memory chips on either side of it. Plugging the chip in backwards will destroy the chip and could damage other components on the board.

Next, verify that the two jumpers on JP6 (next to the new RAM chip) are installed between pins 2 & 3 and pins 4 & 5. Pin 1 of JP6 (the pin closest to the EPROM) should be left open. No jumpers need to be set on the SpectraSense

If you want the log memory to remain intact through a power outage, you'll need to use either a battery-backed RAM or a module that backs up a RAM chip plugged in on top of it (such as the Dallas Semiconductor DS1210 module). If you have any questions about what kind of RAM chip will work in the socket or about how to battery back the memory, contact Circuit Cellar.

When the RAM is installed and the jumpers are set, you may turn the power back on and try data logging.

8.2 Software Setup

The first step that must be performed before logging any data is to clear the memory. Log memory may be cleared in one of two ways: from HOST or from within your XPRESS program.

While in HOST, selecting Clear from the Log menu clears the log memory. Clearing log memory from HOST may be done at any time. You may also embed a ClearLog command within your XPRESS program (see Chapter 7 for more details). Whenever the command is encountered, the log memory is cleared. The command is most often placed at the top of the program and is run as a result of the Reset condition (see Chapter 7). The only potential drawback of clearing log memory on reset is your data is cleared any time you load a new program or the power is lost and restored. If you prefer to have more control over when memory is cleared, stick with HOST's Clear command.

Once the log memory has been cleared, you may start logging data values to memory. Each time a value is saved with the Log command, a 16-bit value, a reference ID number, and the current time and date are saved (for a total of eight bytes). The purpose of the reference ID number becomes clear when you start planning what you want to log.

If data values were saved to memory without IDs, you would be limited to storing data from just one kind of sensor. Otherwise, you wouldn't know which sensor had generated the data. By storing data in memory sequentially with IDs attached to each value, you may sort the data by IDs at a later time. The examples presented later in this section clarify the concepts further.

Up to 4096 samples may be stored in memory at any given time (32768 bytes of RAM divided by 8 bytes per sample). When you reach the end of memory, new samples begin to overwrite the oldest samples, so when you dump the data to a disk file, you end up with the last 4096 samples stored.

When you decide to retrieve the data from the SC's memory, select the Dump command from the Log menu within HOST. The data is stored in a disk file called LOGDATA.BIN on the PC. Be aware that 32K of data transferred from the SC to the PC at 9600 bps takes up to 35 seconds to complete. Be patient.

8.3 What to Log

There are many situations where the time stamp saved with the logged data is more useful to you than the actual data stored. For example, if you want to log the fact that a motion sensor was tripped or a door was opened, simply log a 0 with a reference ID that is unique to the motion sensor or door sensor. When you later process the logged data, you can ignore the data that was saved and associate the reference ID with the time and date saved. Take the following example:

```
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)  =  0
END
```

In the first Log statement, the value of the temperature sensor is stored every 30 minutes. In this case, you use reference ID 0 to find the temperature data in the file, then associate the actual temperature read with the time and date of when it was logged.

In the second two Log statements, you are only interested in when the sensors were tripped, not the value saved. Search the data file for reference IDs of 1, and you'll know every time that the motion sensor was tripped. Search for IDs of 2, and you'll know when the door was opened.

8.4 Logged Data Format

Each logged data sample is stored in the LOGDATA.BIN file as follows:

Byte 0: Reference ID number	(0–254)
Byte 1: Low byte of data	(0–255)
Byte 2: High byte of data	(0–255)
Byte 3: Month logged	(1–12)
Byte 4: Day logged	(1–31)
Byte 5: Hour logged	(0–23)
Byte 6: Minute logged	(0–59)
Byte 7: Second logged	(0–59)

All values are in binary. This file may *not* simply be typed to the screen. It consists of primarily unprintable binary characters and must be further processed on the PC by custom software to be useful. The following section contains some sample processing code written in C to give you an idea of what's necessary.

8.5 Sample Processing Software

The following C program (included on the distribution disk in both source and executable form) may be used to process data logged by the Supervisory Controller and dumped to disk. Feel free to modify the code to fit your needs. It is written in Turbo C, but should work with most C compilers.

When run, the sample code reads the LOGDATA.BIN file (or another file you specify on the command line) and generates a report to the screen with the data grouped by reference ID numbers. The data value, date, and time are displayed, one record per line. You may redirect the output to a file or the printer for a permanent, readable record.

To process the data in the file called MYDATA.BIN, use the command:

```
PROCESS  MYDATA
```

To process the data in LOGDATA.BIN and send it to a file called LOG.TXT, use the command:

```
PROCESS  >LOG.TXT
```

To process the data in the file called WEEKLY.RAW and send it to the printer, use the command:

```
PROCESS  WEEKLY.RAW  >PRN
```

```
#include <stdio.h>
#include <stdlib.h>

#define false 0
#define true  1

FILE *fp
unsigned char *logdata;
unsigned int  len=0;


unsigned int ReadData(char *input)
{
   unsigned int i=0;

   if ((fp=fopen(input, "rb")) == NULL) {
      printf("\nError opening %s\n", input);
      fclose(fp);
      return 1;
   }
   logdata = malloc(32768);
   if (!logdata) {
      printf("\nError allocating memory.\n");
      fclose(fp);
      return 1;
   }
   while (!feof(fp)) {
      logdata[i++] = getc(fp);
      len++;
   }
   fclose(fp);
   return 0;
}

void ProcessData(void)
{
   unsigned int id, record, id_header;

   for (id=0; id<=254; id++) {
      id_header = false;
      for (record=0; record<len; record+=8) {
         if (logdata[record]==id) {
            if (!id_header) {
               printf("\nID = %u\n", id);
               id_header = true;
            }
            printf("   % 5i   %02i/%02i, %02i:%02i:%02i\n",
                   logdata[record+1]+(logdata[record+2]*256),
                   logdata[record+3], logdata[record+4],
                   logdata[record+5], logdata[record+6],
                   logdata[record+7]);
         }
      }
   }
   printf("\n");
}
```

```
main(int argc, char *argv[])
{
   char *file, name[16];
   unsigned int err=0;

   file = name;

   if (argc < 2)
      err = ReadData("LOGDATA.BIN");
   else {
      if (strchr(argv[1],'.'))
         err = ReadData(argv[1]);
      else {
         strcpy(file, argv[1]);
         strcat(file, ".BIN");
         err = ReadData(file);
      }
   }
   if (!err)
      ProcessData();
   return 0;
}
```

Appendix A: Example Programs

Often the best way to learn is by example. The sample programs shown here aren't necessarily intended to do anything useful by themselves, but are meant to show how one or more elements of the XPRESS language are used.

Example 1

```
!
! Blink a direct output at a 2-second rate.
!
CONFIG SC = HCS180

DEFINE BlinkTimer = Timer(0)
DEFINE Light = Output(0)

BEGIN
IF Reset THEN
  BlinkTimer = ON
END

IF BlinkTimer>=2 THEN
  IFA Light=OFF THEN
    Light = ON
  ELSE
    Light = OFF
  END
  BlinkTimer = ON
END
```

Example 2

```
!
! Watch a motion detector connected to a netbit and count
!  how many times motion is detected in a 30-second period.
!  If the count exceeds 5 within that time period, sound
!  the horn also connected to a netbit and turn on the
!  lights.
!
CONFIG SC = HCS180
CONFIG PL-Link = 1
CONFIG DIO = 1
CONFIG LCD = 1

DISPLAY Modules = L

DEFINE Motion       = Netbit(0)
DEFINE MotionCount  = Variable(0)
DEFINE MotionTimer  = Timer(0)
DEFINE Light1       = Module(L5)
DEFINE Light2       = Module(L7)
DEFINE Horn         = Netbit(7)
DEFINE Console      = LCD(0)
```

```
BEGIN
IF Reset THEN
  MotionTimer = OFF; Motion = ON
  Horn = OFF; MotionCount = 0
END

IF MotionTimer>30 AND MotionCount>5 THEN
  Horn = ON
  Light1 = ON; Light2 = ON
  MotionTimer = OFF; MotionCount = 0
  Console = "Motion limit exceeded.\n"
END

IFA Motion=ON THEN
  IF MotionTimer=OFF THEN
    MotionTimer = ON
  END
  IF MotionTimer = ON THEN
    Inc(MotionCounter)
  END
END
```

Example 3

```
!
! Turn on lights in response to motion, but only if it is
!  dark outside. Turn the lights off 20 minutes after the
!  last motion has been detected at all hours (so the
!  lights are automatically turned off even during the
!  day). A light-level detector is connected to an
!  ADIO-Link A/D converter.
!
CONFIG SC = HCS180
CONFIG PL-Link = 1
CONFIG ADIO-Link = 1

DISPLAY Modules = E

DEFINE LivingLamp    = Module(E2)
DEFINE LivingFloods  = Module(E12)
DEFINE LivingTimer   = Timer(45)
DEFINE LivingMotion  = Input(4)
DEFINE LightLevel    = ADC(16)
DEFINE Setpoint      = 85
DEFINE Mode          = Variable(3)
DEFINE Day           = 0
DEFINE Night         = 1

BEGIN
IF Reset THEN
  Mode = Day; LivingTimer = OFF
END
```

```
IF  LightLevel<=Setpoint  THEN
  Mode = Night
END
IF  LightLevel>Setpoint  THEN
  Mode = Day
END
IF  Mode=Night  AND  LivingMotion=EDGE  THEN
  LivingLamp = ON; LivingFloods = ON
END

IF  LivingMotion=EDGE  THEN
  LivingTimer = ON
END

IF  LivingTimer>=20  THEN
  LivingLamp = OFF; LivingFloods = OFF
  LivingTimer = OFF
END
```

Example 4

```
!
! A number of buttons on your hand-held remote are defined
!  to put the HCS-II into different modes. Actions are
!  carried out depending on those modes.
!
CONFIG SC = HCS180
CONFIG PL-Link = 1
CONFIG IR-Link = 3

DISPLAY Modules = B,D,F

DEFINE MovieMode     = IRcode(2)
DEFINE DinnerMode    = IRcode(3)
DEFINE RomanticMode  = IRcode(7)
DEFINE SleepMode     = IRcode(9)
DEFINE FamilyRoom   = 0           ! IR-Link #0
DEFINE DiningRoom   = 1           ! IR-Link #1
DEFINE Bedroom      = 2           ! IR-Link #2
DEFINE FamilyLight   = Module(F3)
DEFINE DiningLight   = Module(D1)
DEFINE BedLight      = Module(B4)
DEFINE FamilyShades  = Output(2)

BEGIN
IF  MovieMode=FamilyRoom  THEN
  FamilyLight = Dim(8)
  FamilyShades = ON
END

IF  DinnerMode=DiningRoom  THEN
  DiningLight = ON
END
```

```
IF  RomanticMode=DiningRoom  THEN
  DiningLight = Dim(10)
END

IF  SleepMode=Bedroom  THEN
  BedLight = OFF; DiningLight = OFF
  FamilyLight = OFF; FamilyShades = ON
END
```

Example 5

```
!
! The following is a simple telephone-based menuing system
!  using the text-to-speech synthesizer for voice prompts
!  and the DTMF board for user input.
!
CONFIG SC = HCS180
CONFIG PL-Link = 1

DISPLAY Modules = A,B

DEFINE LightLevel    = ADC(0)
DEFINE RingTimer     = Timer(3)
DEFINE AlarmOn       = Variable(17)
DEFINE Password      = Variable(30)
DEFINE PhoneLevel    = Variable(31)
DEFINE Selection     = Variable(32)
DEFINE Light         = Variable(33)
DEFINE DoorSensor    = Input(4)
DEFINE InsideLight   = Module(A6)
!=============================================
BEGIN                      ! Start of continuous section

IF Reset THEN
  ClearTimers
  OnHook
  PhoneLevel = 0
  Say"Successful system reset."
END

IF  DoorSensor=EDGE  AND  DoorSensor=ON  THEN
  InsideLight = ON
END

IF  DoorSensor=OFF  THEN
  InsideLight = OFF
END

!=============================================
SEQUENTIAL    ! Start of sequential section
!=============================================
!
! Wait for a single ring followed by a hang up.
! Then wait up to one minute for another ring.
! If detected, immediately pick up the phone.
! This is done so an answering machine can be
! used on the same line as the computer.
!
IF RingTimer=OFF AND Rings=1 THEN
  RingTimer = ON
END

IF RingTimer=ON AND Rings>1 THEN
  RingTimer=OFF
END
IF RingTimer>60 THEN
  RingTimer = OFF
END

IF RingTimer>8 AND Rings=1 THEN
  RingTimer = OFF
  OffHook
  Say"~@ ~60P ~3F"
  Say"~2 Hello. ~5S"
  Say"Welcome to our house."
  Say"Please enter your password."
  Password = DTMFnumber(150)
  IFA Password=1234 THEN
    Say"|Thank you."
    PhoneLevel = 1
  ELSE
    SayW"That is wrong. Goodbye."
    PhoneLevel = 0
    OnHook
  END
END

IFA PhoneLevel=1 THEN
  Say"Please make a selection."
  Say"Press zero for a list of choices."
  Selection = DTMFdigit(250)
  IFA Selection<0 THEN
    SayW"Timed out. Good bye."
    PhoneLevel = 0
    OnHook
  END
  IFA Selection=0 THEN
    Say"|Press 1 for light level.."
    Say"Press 2 for alarm status.."
    Say"Press 9 to finish.."
  END
  IFA Selection=9 THEN
    SayW"|Have a nice day. Good bye."
    PhoneLevel = 0
    OnHook
  END
  IFA Selection=1 THEN
    Light = (LightLevel * 100) / 26
    Say"|~D DH IY ~T outside light"
    Say"is at %P1%%.",Light
  END
```

```
   IFA Selection=2 THEN
     IFA AlarmOn=TRUE THEN
       Say"|~D DH IY ~T alarm is on."
     ELSE
       Say"|~D DH IY ~T alarm is off."
     END
  END
   IFA (Selection>2 AND Selection<9)
       OR Selection>9 THEN
     Say"|Invalid selection. Please try again."
  END
END

IF PhoneLevel=0 THEN
  OnHook
END
```

Example 6

```
!
! Allow both a modem and the voice/DTMF pair to occupy the
!  same phone line.  If you call and allow the phone to ring
!  twice, the voice is used to answer the phone and presents
!  you with a list of options.  If you call, let the phone
!  ring once, hang up, then call again within 1 minute, the
!  modem will answer instead, allowing the remote use of
!  HOST.
!
CONFIG SC = HCS180

DEFINE RingTimer = Timer(3)

BEGIN

IF Reset THEN
   ClearTimers
  ModemRings = 0
END

IF RingTimer=OFF AND Rings=1 THEN
  RingTimer = ON
END

IF RingTimer=ON AND Rings>1 THEN
  RingTimer = OFF
  ModemRings = 0                    ! Disable modem answer
END

IF RingTimer>60 THEN
  RingTimer = OFF
  ModemRings = 0                    ! Disable modem answer
END
```

```
IF RingTimer>8 THEN
  ModemRings = 1                    ! Enable modem answer
END

SEQUENTIAL

IF Rings>=2 THEN
   RingTimer = OFF
   OffHook
   SayW"This is the menu"
   .
   .
   .
   OnHook
END
```

Appendix B: Port Assignments

To maintain ease of configuration and setup, we have preassigned port numbers to fixed hardware arrangements. Ports with fixed assignments include all direct inputs and outputs, netbits, A/D converters, and D/A converters. There is no way to reassign port numbers, nor is there any way to reassign input/output assignments. Netbits on DIO-Links may be either inputs or outputs depending on whether the bits are read or written. All other ports are fixed as either input or output. The current state of all digital outputs may be tested within IF/IFA statememts.

Direct I/O Ports

The direct digital I/O ports are referenced using the Input and Output keywords in XPRESS.

Board	Address	I/O	Number	Connector	Board Labels
HCS180/	8000	In	0–7	J4	PA0–PA7
HCS2-DX	8001	In	8–15	J4	PB0–PB7
	8002	Out	0–7	J4	PC0–PC7
BUFIO/	A000	In	16–23	J1	IA0–IA7
BUF50	A001	In	24–31	J2	IB0–IB7
	A002	In	32–39	J3	IC0–IC7
	A010	Out	16–23	J4	OA0–OA7
	A011	Out	24–31	J5	OB0–OB7
	A012	Out	32–39	J6	OC0–OC7
	A020	In	184–191	J1	IA0–IA7
	A021	In	192–199	J2	IB0–IB7
	A022	In	200–207	J3	IC0–IC7
	A030	Out	184–191	J4	OA0–OA7
	A031	Out	192–199	J5	OB0–OB7
	A032	Out	200–207	J6	OC0–OC7
IND180	C800	In	40–47	J5	PA0–PA7
	C801	In	48–55	J5	PB0–PB7
	C802	In	56–63	J5	PC0–PC7
	C900	Out	40–47	J6	PA0–PA7
	C901	Out	48–55	J6	PB0–PB7
	C902	Out	56–63	J6	PC0–PC7
IND54	EB00/EF00	I/O	64–71	J4	4PA0–4PA7
	EB01/EF01	I/O	72–79	J4	4PB0–4PB7
	EB02/EF02	I/O	80–87	J4	4PC0–4PC7
	EA00/EE00	I/O	88–95	J3	3PA0–3PA7
	EA01/EE01	I/O	96–103	J3	3PB0–3PB7

EA02/EE02	I/O	104–111	J3	3PC0–3PC7
E900/ED00	I/O	112–119	J2	2PA0–2PA7
E901/ED01	I/O	120–127	J2	2PB0–2PB7
E902/ED02	I/O	128–135	J2	2PC0–2PC7
E800/EC00	I/O	136–143	J1	1PA0–1PA7
E801/EC01	I/O	144–151	J1	1PB0–1PB7
E802/EC02	I/O	152–159	J1	1PC0–1PC7

The HCS-DTMF telephone interface board has four LEDs, two of which may be controlled from your XPRESS program. Reference Output(8) to control LED3, and Output(9) to controle LED4.

The direction of the data flow of the ports on the IND54 expansion board is determined by the port address jumper settings on the board (as shown above). For example, to use the 24 bits on connector J1 as inputs 136–159, set the jumpers on the board to select a base address of E800 for J1. To use the bits as outputs 136–159, set the base address to EC00 instead.

Netbits

Netbits are digital I/O bits found on the DIO-Link, DIO+-Link, LCD-Link, MCIR-Link, and AMAN-Link modules. Up to eight DIO-Links, eight DIO+-Links, eight LCD-Links, and eight AMAN-Links may be used on the same network. The Netbit XPRESS keyword is used to both set and read netbits.

DIO-Link netbits may be either inputs or outputs. To use a bit as an output, simply set the bit to either 1 or 0. To use the bit as an input, it must be set to a 1 on reset, then the bit may be read at any time and the current state of the input will be returned.

LCD-Link netbits are fixed as inputs. The LCD-Link returns only a change in the input’s state from 0 to 1, so are best used with push buttons.

DIO+-Link and AMAN-Link netbits are fixed as either inputs or outputs.

Only the A and B outputs of the MCIR-Link are supported.

DIO0: 0–7
DIO1: 8–15
DIO2: 16–23
DIO3: 24–31
DIO4: 32–39
DIO5: 40–47
DIO6: 48–55
DIO7: 56–63

LCD0: 64–67
LCD1: 68–71
LCD2: 72–75
LCD3: 76–79
LCD4: 80–83
LCD5: 84–87
LCD6: 88–91
LCD7: 92–95

AMAN0: 96–103 (Input)	104–111 (Output)
AMAN1: 112–119(Input)	120–127 (Output)
AMAN2: 128–135 (Input)	136-143 (Output)
AMAN3: 144-151 (Input)	1152-159 (Output)
AMAN4: 160-167 (Input)	168-175 (Output)
AMAN5: 176-183 (Input)	184-191 (Output)
AMAN6: 192-199 (Input)	200-207 (Output)
AMAN7: 208-215 (Input)	216-223 (Output)

MCIR0: 240,241 (Output)
MCIR1: 242,243 (Output)
MCIR2: 244,245 (Output)
MCIR3: 246,247 (Output)
MCIR4: 248,249 (Output)
MCIR5: 250,251 (Output)
MCIR6: 252,253 (Output)
MCIR7: 254,255 (Output)

DIOP0: 256–259 (Input)	260–263 (Output)
DIOP1: 264–267 (Input)	268–271 (Output)
DIOP2: 272–275 (Input)	276–279 (Output)
DIOP3: 280–283 (Input)	284–287 (Output)
DIOP4: 288–291 (Input)	292–295 (Output)
DIOP5: 296–299 (Input)	300–303 (Output)
DIOP6: 304–307 (Input)	308–311 (Output)
DIOP7: 312–315 (Input)	316–319 (Output)

Netbytes

Netbytes are identical to netbits but are grouped by eight to allow bytewide testing and setting of network I/O bits. All of the following netbytes may be read, tested, and assigned to variables, but only the first eight (corresponding to the DIO-Links) may be set (changed).

Netbyte	Link	Netbits	Netbyte	Link	Netbits
0	DIO0	0–7	21	MAN4	168–175
1	DIO1	8–15	22	MAN5	176–183
2	DIO2	16–23	23	MAN5	184–191
3	DIO3	24–31	24	MAN6	192–199
4	DIO4	32–39	25	MAN6	200–207
5	DIO5	40–47	26	MAN7	208–215
6	DIO6	48–55	27	MAN7	216–223
7	DIO7	56–63	28	N/A	224–231
8	LCD0,1	64–71	29	N/A	232–239
9	LCD2,3	72–79	30	MCIR0–3	240–247
10	LCD4,5	80–87	31	MCIR4–7	248–255
11	LCD6,7	88–95	32	DIOP0	256–263
12	MAN0	96–103	33	DIOP1	264–271
13	MAN0	104–111	34	DIOP2	272–279
14	MAN1	112–119	35	DIOP3	280–287
15	MAN1	120–127	36	DIOP4	288–295
16	MAN2	128–135	37	DIOP5	296–303
17	MAN2	136–143	38	DIOP6	304–311
18	MAN3	144–151	39	DIOP7	312–319
19	MAN3	152–159			
20	MAN4	160–167			

A/D Converters

The system A/D converters accept analog voltages in and produce digital results, the range of the output values depending on the resolution of the A/D converter. The ADC XPRESS keyword is used to read the converter channels. The SC's ADC is continuously scanned, resulting in each channel being read several times per second. The network ADCs are read only every few seconds. The frequency depends on how many modules are on the network.

HCS180/HCS2-DX: 0–7

IND13: 8–15	IND30: 96–111
MAN0: 16–21	DIOP0: 112,113
MAN1: 24–29	DIOP1: 114,115
MAN2: 32–37	DIOP2: 116,117
MAN3: 40–45	DIOP3: 118,119
MAN4: 48–53	DIOP4: 120,121
MAN5: 56–61	DIOP5: 122,123
MAN6: 64–69	DIOP6: 124,125
MAN7: 72–77	DIOP7: 126,127

D/A Converters

While a number is assigned to the optional second ADC channel on the DIO+-Link boards, only one 8-bit ADC channel per board is supported.

The IND13 should be addressed for 9000H and the IND30 should be addressed for 9800H. Only one ADC board (IND13 or IND30) may be installed in an industrial (IND) system.

The only D/A converters in the system are on AMAN-Link modules. Any value from 0 to 255 may be sent to one of eight modules using the DAC XPRESS keyword, and the corresponding channel outputs a voltage from 0 to 5 volts. Note that DAC functionality requires optional DAC chips on the AMAN-Link board.

MAN0: 0–1	MAN4: 16–17
MAN1: 4–5	MAN5: 20–21
MAN2: 8–6	MAN6: 24–25
MAN3: 12–13	MAN7: 28–29

Appendix C: What's New in Version 3

Users familiar with the features in previous versions of the HCS-II software don't need to spend a lot of time rereading the entire manual. The following list of new features covers the highlights of what is new in this version. Details for each of the new features may be found elsewhere in this manual (generally in Chapters 6 and 7). Also be sure to review the new HOST features in Chapter 4 and the new modem support in Chapter 5.

XPRESS programs written for Version 2 of the software may be used as is with Version 3 with one exception. Defining the hardware using the CONFIG command has been changed to be more flexible. See the description of CONFIG in Chapter 7 for more details. No other aspect of the existing language has been changed.

✔ New HCS/PC interface
- Query and override any system input, output, or parameter from the PC

✔ Revamped HOST program
- Use a mouse (or the keyboard) to resize, move, open, and close all HOST windows
- Netbits and analog outputs are now also displayed on the HOST screen
- Send messages from XPRESS to a HOST window for debugging
- Set or clear system inputs, outputs, or X-10 modules and try out speech strings from HOST

✔ Modem support
- Call your HCS-II from a remote location and check its status, load a new XPRESS program, or retrieve logged data

✔ Caller ID
- Access Caller ID data from XPRESS to announce or log who's calling

✔ Support for more digital I/O expansion boards, AnswerMAN Link module and HCS-PLIX

✔ Read and write eight netbits at a time with the Netbyte command

✔ Send commands to network modules directly from HOST and XPRESS

✔ Detect loss of AC power from within XPRESS

✔ Send commands to receive-only network modules from XPRESS without having to define the modules at the top of your XPRESS program

✔ Check the amount of data currently logged from HOST and XPRESS

- ✔ LEDs on the HCS-DTMF board and the SpectraSense are now assigned output numbers so they can be controlled from XPRESS
- ✔ MCIR-Link digital outputs are now assigned netbit numbers so they can be controlled from XPRESS
- ✔ The compiler version number is embedded in the compiled binary code and is compared against the firmware version on a program load to ensure a match

Except for the modem features, no additional hardware is required.

Upgrade Instructions

Upgrading a previous-version HCS-II is as easy as replacing the EPROM on the Supervisory Controller and switching to the new COMPILE and HOST programs on the PC. Recompile your XPRESS program, reload it into the SC using HOST, and your system is back on-line.

The following steps should be taken to install the Version 3 upgrade:

- ✔ Turn off power to the Supervisory Controller. Changing EPROMs with power still on may damage the board and the EPROM.
- ✔ Carefully remove the previous-version EPROM from socket U9 on the SC (or socket U10 on an IND180 or socket U13 on a SpectraSense). It is often easiest to insert a small screwdriver on one end between the socket and the chip (*not* between the socket and the board!) and gently rock the chip out of the socket.
- ✔ Install the new EPROM in the same socket. Be sure pin 1 of the EPROM is closest to the edge of the board. The chip should be oriented in the same way as the chip you just removed. Take special care not to bend any pins on the chip while inserting it into the socket. (On the SpectraSense, make absolutely sure the EPROM is bottom justified in the socket. In other words, there should be four empty pins in the socket at the same end as the notch. When in doubt, the EPROM should be oriented in its socket in exactly the same way as the RAM chip in socket U15.)
- ✔ Reapply power to the Supervisory Controller board. Your old XPRESS program is automatically cleared, so the system shouldn't be doing much right now.
- ✔ Check to be sure you are connecting your PC to the SC with a serial cable that passes all the signals. That is, if your cable only passes pins 2, 3, and 7, you must modify it to add at least pins 4, 5, 6, and 20. If that is a problem, connect pins 4 and 5 and pins 6 and 20 together at the PC end instead. The new HOST

program is much more picky about the state of the serial port handshaking lines on the PC side.

✔ Locate the new distribution disk. Replace your old COMPILE and HOST programs with the new versions found on the disk. The old PC software may *not* be used with the new SC EPROM.

✔ Change the SC definition at the top of your XPRESS program to reflect the processor board you're using. If you currently set the SC type in the CONFIG statement to SC1, change it to HCS180. If you have some other hardware, refer to the CONFIG command in Chapter 7 for more details. No other changes should be necessary. Recompile your program with the new COMPILE.

✔ Run the new HOST. Press "F" to select the file menu and "L" to load your recompiled XPRESS program into the SC. The system should start running the same way it did before the upgrade. If you get timeout errors, recheck your serial connections as described a few steps back.

✔ Start experimenting with the new features and have some fun.

Appendix D: HOST/SC Serial Protocol

The primary elements of the HCS/SS are the Supervisory Controller (SC) and the host IBM PC-compatible computer. The SC is responsible for overall system control while the host computer is used to edit and compile XPRESS programs, send the programs to the SC, set a new time in the SC, clear log memory, receive logged data and dump it to a disk file, and display system status. The program on the host computer used to communicate with the SC is called HOST.

The HOST/SC interface allows a great deal of control over the system from the PC side of the connection. The SC sends status information to the PC for display on the console at periodic intervals. What information is sent by the SC can be controlled from the PC. The PC may also send the SC a one-time request for the status of any system component and override that status or set a new value.

Not all elements of the interface are used by the stock HOST program. The interface is fully defined to allow third parties to develop custom PC interfaces to the HCS/SS to better fit individual installations.

Command Summary

All commands sent from the PC to the SC are of the form:

```
! <cmd> <data>
```

where <cmd> is the command to be executed and <data> is an optional data field of variable length, depending on the command.

Most replies are of the form:

```
$ <cmd> <reply>
```

where <cmd> is the command generating the reply and <reply> is an optional data field of variable length, depending on the command. Two commands don’t generate any reply, while one other uses a special reply sequence.

The following table describes the commands and their replies:

cmd	Description	data	reply	
00	Reset system (same as hardware reset)	none	—	no reply
01	Pause all status output until next command	none	none	
02	Request full system status information	none	—	no reply
03	Select time display frequency	a	none	a=0:off, 1:sec 2:min
04	Get time once	none	a–h	a–h=time data
05	Set new time	a–h	none	a–h=time data
06	Load new XPRESS program	prog	—	special timing used
07	Clear log memory	none	none	
08	Get size of logged data	none	a,b	a=low, b=high (0–4095)
09	Get logged data	none	block	terminated with FF,FF,FF
0A	Put data in log memory	a,b,c	none	a=ID, b=low, c=high
10	Select X-10 display	a,b	none	a=A–H, b=I–P
11	Get X-10 status once	a	a,b	a=house/mod, 0–F=A–P/1–16 b=0:off, 1:on
12	Set X-10 module	a,b,c	none	a=house/mod, b=0:alloff, 1:allon, 2:on, 3:off, 4:dim, 5:bright c=dim/br count
13	Select dig in display	a,b,c,d	none	a–d=bitmap
14	Get dig in status once	a	a,b	a=input, b=0:off, 1:on
15	Set dig in state	a,b	none	a=input, b=0:off, 1:on, 2:transparent
16	Select dig out display	a,b,c,d	none	a–d=bitmap
17	Get dig out status once	a	a,b	a=output, b=0:off, 1:on
18	Set dig out state	a,b	none	a=output, b=0:off, 1:on
19	Select ADC display	a,b,c	none	a–c=bitmap
1A	Get ADC value once	a	a,b,c	a=ADC, b=low, c=high

cmd	Description	data	reply	
1B	Set ADC value	a,b,c	none	a=ADC, b=low, c=high (0–4095,4096= transparent)
1C	Select DAC display	a	none	a=bitmap
1D	Get DAC value once	a	a,b	a=DAC, b=value
1E	Set DAC value	a,b	none	a=DAC, b=value
1F	Select net module display	a	none	a=bitmap
20	Get net module status once	a	a,b	a=mod/num, b=0:timeout, 1:active, 2:err
21	Select netbit display	a,b,c,d,e	none	a-e=bitmap
22	Get netbit status once	a,b	a,b,c	a=low, b=high, c=0:off, 1:on
23	Set netbit state	a,b,c	none	a=low, b=high, c=0:off, 1:on
24	Get variable value once	a	a,b,c	a=variable, b=low, c=high
25	Set variable value	a,b,c	none	a=variable, b=low, c=high
30	Send string to network	string	none	zero terminated
31	Send string to voice	string	none	zero terminated

Status Display Summary

The HCS/SS periodically sends status information to the PC. What information is sent depends on what has been requested by the PC using the commands described above. The format of the display packet is the same as the reply packet desribed above, namely:

$ <cmd> <data>

where <cmd> indicates what kind of information is being sent sent and <data> contains the actual status information. The following table summarizes the information sent by the SC to the PC.

cmd	Description	data	
80	Current time	a–h	a–h=time data
81	X-10 modules	a,b,c	a=house, b=low, c=high
82	Digital inputs	a,b	a=group, b=data
83	Digital outputs	a,b	a=group, b=data
84	Analog inputs	a,b–q	a=group, b–q=data low, high
85	Analog outputs	a,b–e	a=group, b–e=data

cmd	Description	data	
86	Network modules	a,b	a=mod/num, b=0:timeout, 1:active, 2:error
87	Netbits	a,b	a=group, b=data
88	Console message	string	zero terminated

Data Formats

Data is formatted as efficiently as possible to minimize the number of bytes being sent over the interface. The following describes various data formats mentioned in the above tables:

Time Data

All time references consist of eight BCD bytes in the order hundredths of second (0–99), second (0–59), minute (0–59), hour (0–23), day of week (1–7, 1=Sunday), day (1–31), month (1–12), year (00–99).

X-10 Data

When described as *house/mod*, the module identifier is packed into a single byte. The upper nybble contains the house code (0–F = A–P) and the lower nybble contains the module number (0–F = 1–16). When selecting X-10 modules to display, the 16 house codes are bitmapped into two bytes. The low-order byte is first, with A in the least-significant bit.

In the status display, the house code being sent by the SC is indicated by an ASCII letter in the range of “A”–“P”. The 16 modules within the housecode are packed into two bytes—low-order byte first—with module 1 in the least-significant bit.

Digital Input Bitmap

The digital inputs are grouped into blocks of eight. When selecting which inputs to display, those blocks are represented as four bitmapped bytes as follows:

56–63	48–55	40–47	32–39	24–31	16–23	8–15	0–7
7	6	5	4	3	2	1	0

120–127	112–119	104–111	96–103	88–95	80–87	72–79	64–71
15	14	13	12	11	10	9	8

184–191	176–183	168–175	160–167	152–159	144–151	136–143	128–135
23	22	21	20	19	18	17	16

248–255	240–247	232–239	224–231	216–223	208–215	200–207	192–199
31	30	29	28	27	26	25	24

When the status information is sent to the PC for display, the group number is sent first (as a single byte), followed by the status of the eight inputs in that group bitmapped into a single byte. The lowest numbered input in the group is represented in the least-significant bit of the status byte.

For example, to indicate the status of inputs 224–231, the SC sends a 29 followed by a byte containing the status of those eight inputs.

When selecting which inputs to display, the same four bitmapped bytes are used. For example, to select inputs 24–39 and 104–111 to be displayed, send the SC the byte sequence 18h, 20h, 00h, 00h.

Digital Output Bitmap

Digital outputs are handled the same way as digital inputs.

Analog Input Bitmap

Analog input selection is handled the same way as digital inputs, but is limited to the first three bitmap bytes (i.e., analog channels 0–191 are supported by the bitmaps, though only 0–135 do anything).

In the status packet sent to the PC, the group number is sent first, followed by 16 bytes of data (eight 16-bit channels, low byte first, low channel first).

Analog Output Bitmap

Analog output selection is handled the same way as the digital inputs, but is limited to the first bitmap byte (i.e., analog channels 0–63 are supported by the bitmaps, though only 0–31 do anything).

In the status packet sent to the PC, the group number is sent first, followed by 4 bytes of data (four 8-bit channels, low channel first).

Netbits

Netbits are handled the same way as digital inputs, but add another bitmap byte as follows:

312–319	304–311	296–303	288–295	280–287	272–279	264–271	256–263
39	38	37	36	35	34	33	32

Network Modules

Network module types are always referenced by the following numbers:

0:PL	4:ADIO
1:MCIR	5:DIO+
2:LCD	6:reserved
3:DIO	7:reserved

When selecting which modules to display, the numbers correspond to bits in a byte. Setting a particular bit selects that module type for display. For example, sending a selection byte of 26h enables the display of the status of all MCIR-Links, LCD-Links, and DIO+-Links in the system.

When sending the network module status information, the SC includes the module type in the upper nybble of the first byte and the module number in the lower nybble. The actual status follows in the second byte. For example, to indicate that DIO-Link number 2 is active on the network, the SC sends 32h, 01h.

New Program Load

When sending a new XPRESS program to the SC, a special handshaking sequence is used. After the initial "!" 06h command is sent, the HOST program must wait for a "^" acknowledge character. This acknowledgement tells the PC that the SC has successfully shut down processing of the old XPRESS program and is ready to accept a new one. At this point, HOST sends the entire compiled XPRESS program. After the last byte has been sent, HOST waits for another acknowledgement character. This time the character indicates the final status of the program load as follows:

^ = Successful program load
= Firmware version doesn't match compiler version
@ = Program too big to fit in SC memory

If either of the latter two errors occurs, the SC clears its memory so there is no XPRESS program there at all. There is no way to get the old program back after the command has been initiated.

XPRESS 3.6 Addendum

Version 3.6 of the HCS II firmware and XPRESS language primarily adds support for the Answer MAN network module. The Answer MAN offers many interfacing options such as digital and analog I/O, frequency in, totalizer in, PWM waveform generation, iButton serial number reading, and LCD display and keypad support. Note that not all options are available simultaneously and some may require additional hardware.

Upgrading

Upgrading your HCS II is simply a matter of installing the new EPROM, recompiling your XPRESS program with the new compiler, and loading your recompiled program into the HCS II. The version numbers of the EPROM and the compiler must match or the HCS will refuse to accept your program upload. Therefore, it is important to remove all old copies of the XPRESS compiler when upgrading.

Configuring the Answer MAN

The Answer MAN module must be configured before it can be used with an HCS II network. It is not possible to configure an Answer MAN from the HOST screen or while it is installed in the HCS network. The Answer MAN must be removed from the network, connected to a PC or serial terminal, put into configuration mode, and set up directly. Please refer to the Answer MAN datasheet for more information on configuring the module.

When using the LCD and keypad support of the Answer MAN, be sure to configure the module address as TERM0, TERM1, and so forth up to TERM7. The TERM*x* address is the default network address of the LCD-Link. The Answer MAN is designed to emulate the LCD-Link directly, so your XPRESS program should access the Answer MAN as if it were an LCD-Link.

When using any of the other Answer MAN features, be sure to address the module as MAN0, MAN1, on up to MAN7. The HCS automatically polls each Answer MAN to find out how it is set up and will respond to only those features for which the module is configured. Note that it may take up to a minute for the HCS to poll the Answer MAN for its setup and to adapt to that setup.

Finally, in all cases, be sure to issue a CD 32 command to set a reply delay of 50 ms.

ADIO-Link and Channel Numbers

ADIO-Link support is removed to make way for Answer MAN support. ADIO-Link netbits are redefined as shown in the following table. Netbits and analog I/O are accessed using the Netbit(), ADC(), and DAC() XPRESS commands. See the *XPRESS Reference Manual* for more details on the use of those commands.

Answer MAN Number	Input Netbits	Output Netbits	Analog Input Channels	Analog Output Channels
0	96–103	104–111	16–21	0,1
1	112–119	120–127	24–29	4,5
2	128–135	136–143	32–37	8,9
3	144–151	152–159	40–45	12,13
4	160–167	168–175	48–53	16,17
5	176–183	184–191	56–61	20,21
6	192–199	200–207	64–69	24,25
7	208–215	216–223	72–77	28,29

Analog inputs and outputs are likewise redefined. Note that only six analog inputs are supported on each Answer MAN, but eight slots have been reserved. The highest two analog input numbers on each module are not used at this time. Similarly, only two analog outputs are supported on each module, but four slots each have been reserved.

XPRESS Configuration

Your XPRESS program must be modified slightly to inform the Supervisory Controller that one or more Answer MAN modules are present on the network. At the top of your program, include the following line:

Config AMAN-Link = *xx*

where *xx* is the number of Answer MAN modules you have on the network.

New Commands

While most of the Answer MAN features can be accessed using existing XPRESS commands, there are several new commands that take advantage of some of the modules new features. The following section describes these new commands.

PWMtotal — Set the total PWM period value

Syntax: PWMtotal(n) = t

where: n = Answer MAN module number (0–7)
t = total period count (0020–7FFF)

See the Answer MAN SP command for details on how to calculate values. Setting this value to zero shuts off the PWM output.

Set the high PWM period value — PWMhigh

Syntax: PWMhigh(n) = h

where: n = Answer MAN module number (0–7)
h = high period count (0008–7FF0)

See the Answer MAN SP command for details on how to calculate values. Setting this value to zero shuts off the PWM output.

Period of frequency being fed to Answer MAN module — Frequency

Syntax: f = Frequency(n)

where: f = period of input frequency (0–32767)
n = Answer MAN module number (0–7)

To calculate frequency from the period returned, use the following equation:

11184 / (Frequency(1) / 10)

Number of pulses received by the Answer MAN since last clear — Total

Syntax: t = Total(n)

where: t = total number of pulses received (0–32767)
n = Answer MAN module number (0–7)

Note that the count is *not* cleared when read. Use ClearTotal to clear the count.

Clear the Answer MAN totalizer count — ClearTotal

Syntax: ClearTotal(n)

where: n = Answer MAN module number (0–7)

Configure the I/O port bit directions — BitDirection

Each Answer MAN digital I/O bit can be set up as either an input or an output. Each of the eight I/O bits corresponds to a bit in a direction byte. When a bit in the direction byte is set to 0, its I/O bit is configured as an output. When a bit in the direction byte is set to 1, the I/O bit is set up as an input. This command is used to configure the directions of all eight bits.

Note that since the HCS doesn't support hexadecimal numbers, the direction byte must be converted to decimal to be used in an XPRESS program.

Syntax: BitDirection(n) = b

where: n = Answer MAN module number (0–7)
b = decimal equivalent of the hexadecimal value that defines each bit as an input or output

KeyDigit   Read a single keypad press

When the Answer MAN module is configured as an LCD-Link (module name TERMn), it supports a scanned matrix keypad. The Answer MAN will buffer up to eight keypresses, so the user can press the keys quickly without losing any data. The KeyDigit function is used to get the next keypress from the buffer.

The KeypadTimeout function is used to set how long the KeyDigit function will wait for a keypress. A negative value is returned if no key is available before the end of the timeout period.

Only works in the sequential section.

Syntax: KeyDigit(n)

where: n = Answer MAN module number (0–7)

The value returned is 0–9 (0–9), 10 (*), 11 (#), or 12–15 (A–D).

KeyNumber   Read up to four keypad presses as a number

When the Answer MAN module is configured as an LCD-Link (module name TERMn), it supports a scanned matrix keypad. The Answer MAN will buffer up to eight keypresses, so the user can press the keys quickly without losing any data. The KeyNumber function is used to get up to four keypresses from the buffer and returns them as a single large numeric value. The function waits until four keys have been pressed or a nonnumeric key is pressed (for fewer than four keypresses).

The KeypadTimeout function is used to set how long the KeyNumber function will wait for a keypress. A negative value is returned if no key is available before the end of the timeout period. Each keypress restarts the timeout timer, so it's not necessary to press all four keys within a single timeout period.

Only works in the sequential section.

Syntax: KeyNumber(n)

where: n = Answer MAN module number (0–7)

The value returned ranges 0000–9999.

Set the timeout value for a keypad read — KeypadTimeout

When the keypad is read, sequential section processing pauses until one or more keypad keys are pressed or a timeout occurs. This function is used to set the timeout for the keypad functions.

Only works in the sequential section.

Syntax: KeypadTimeout(n) = t

where: n = Answer MAN module number (0–7)
t = timeout value in hundreds of milliseconds (0–255 = 0 to 25.5 seconds)

Read a Dallas Semiconductor iButton serial number — iButton

All Dallas Semiconductor iButton devices include a unique serial number that may be read independent of any other functions the device supports. The Answer MAN may be set up to read this serial number and pass it along to the HCS.

The byte sequence consists of an iButton family code, followed by several bytes of the serial number, followed by several zeros, and terminated with a CRC. If the byte sequence printed on the device consists of a single byte followed immediately by several zeros, reverse the order of the bytes before using the sequence.

When the Answer MAN detects an iButton device, it reads the serial number and stores it in a buffer. The HCS then polls the Answer MAN for that number and stores it in its own buffer. If a new serial number comes in before the original one was tested by an XPRESS program, the new number overwrites the old one.

Note that iButton values may only be tested within an IF statement and must be compared to constants. iButton serial numbers may *not* be stored in variables nor may they be compared to values stored in variables. However, an iButton serial number remains available for testing until a new value is read. That is, it may be tested multiple times within an XPRESS program without being automatically cleared by the system.

Syntax: iButton(n) = s1,s2,s3,...,s8

where: n = Answer MAN module number (0–7)
s1–s8 = eight decimal bytes of an iButton serial number

Example:
```
DEFINE FredKey = 2,70,242,0,0,0,0,152
DEFINE GeorgeKey =

BEGIN
IF iButton(0)=FredKey THEN
  Console = “Fred’s key detected”
END
IF iButton(0)=GeorgeKey THEN
  Console = “George’s key detected”
END
```

Answer MAN Versions

The majority of features of all Answer MAN modules since V 1.0 work fine with XPRESS 3.5. The exception is 12-bit analog I/O. You must be using V 1.11 or later of the Answer MAN in order to use 12-bit analog I/O with the HCS.

To find out what version Answer MAN you have, connect a terminal to the Answer MAN, ground the CFG pin, and power it up in configuration mode. The signon banner displayed on reset indicates the Answer MAN version number.

XPRESS v3.64 & v4.01 Addendum

v3.64 and v4.01 of XPRESS have added some new functionality to commands that send strings to modules.

Commands Affected: Console, LCD(), LPT(), SAY, SAYW

The above commands send strings to their respective modules. These strings can contain '%' flags to include real-time data from the system or variables in the string when it is sent. The following flags have been added to this release of XPRESS:

%M – If a Caller-ID modem is connected to the HCS-II controller and it is enabled, %M will output the last phone number received from Caller-ID.
%N – If a Caller-ID modem is connected to the HCS-II controller and it is enabled, %N will output the last name received from Caller-ID.

You must subscribe to Caller-ID and have a functioning Caller-ID capable modem connected to your HCS-II for the above flags to work. Otherwise they will output nothing when used.

Example:

```
IF CIDNew = TRUE THEN
   LPT(0) = "%N called from %M\n"
END
```

Would output the following on printer LPT(0):

```
Joe Smith called from 123-456-7890
```

Note that the name displayed exactly as a Caller-ID unit would display it. Some telephone companies may send the last name first and the HCS-II will output it that way.

%Sx – This flag is used to display a value stored in any XPRESS variable. The value is output with x digits using leading zeros unless the value has more than x digits. If the value has more than x digits, it will be output entirely with no leading zeros.

Example:

```
Define Call_Minute = Variable(0)
Define Call_Hour   = Variable(1)

BEGIN

IF CIDNew = TRUE THEN
   Call_Minute = CIDminute
   Call_Hour = CIDhour
   LPT(0) = "%N called at %P0:%S2\n", Call_Hour, Call_Minute
END
```

If the call arrived at 2:05, the above code would print:

```
Joe Smith called at 2:05
```

If you used %P0 instead of %S2, the time would be displayed as "2:5".