HSI-24/14 OPERATOR'S MANUAL VERSION 3.0

Probe Products Corporation

1763 Baseline Road

Grand Island, NY 14072

(716) 773-5554

(716)-773-5336 FAX

Table of Contents

The HSI-24, an Overview

Probe Product Corporation's HSI-24 is a high speed signal conditioning/data processing system designed to allow direct connection of gaging and displacement transducers to an IBM PC, XT, AT ,386, 486 or compatible computer.

To acquaint a new user with the powerful features of the HSI-24, an application program is furnished with the system. This program can be used "as-is" to set up a measurement system, or it can be used as a basis for customized software. Source code for this application program is included as part of the system.

System Components

The system consists of four parts:
1- a master system board with onboard 16-bit processor, A/D converter and all other active analog and digital devices, residing within the computer and occupying one slot,
2- a cable assembly to connect the system board to a passive external junction box with transducer sockets,
3- a passive junction box (normally located near the transducers) with sockets suitable for the transducers in use, and
4- a diskette containing the internal operating software (firmware) for the system and sample application/setup programs.

Some systems may contain special cable assemblies for specific applications, or no cable and junction box at all, when those connections are to be made on-site.

Large Systems

For those applications requiring conditioning for more than 24 transducers, slave system boards, cable assemblies and boxes are added to the master system. Up to 96 transducers can be conditioned by one such master/slave system.

Auxiliary Inputs

Besides the transducer inputs, each system board (master or slave) has four +/-5VDC inputs which may be used for any analog signal which has (or can be modified to have) suitable values. Digital inputs from TTL devices, Hall-effect proximity switches and switch closures are typical inputs which can also be used.
A fully expanded HSI-24 system will, therefore, have the capacity for 96 transducers and 16 voltage sources.

High Speed Operation

It is important to note that all signal conditioning for the transducers is accomplished within the system--no external gaging amplifiers, A/D converters or communication ports are required.

The HSI-24 lends itself to dynamic gaging, since complex functions can be completed by the on-board co-processor, with results being passed to the host computer. And, since all communications to the host computer are passed directly to its bus, data transfer rates are many times faster than those possible using RS-232, RS-422 or IEEE- 488 interfaces. These two factors combine to allow very high speed gaging operations. Transducer signals, or even complex measurement functions (using the unique formulae construction techniques) can be passed to the host at up to 2,500 readings per second.

Quick Start

For those users who are sure they are familiar with the installation of adapter cards in PC type equipment, this section provides a brief setup procedure.

1. Remove power and cover from PC.
2. Install HSI-24 card. Be sure to install the rear bracket hold-down screw, otherwise the force from flexing the large interface cable will pop the HSI-24 card out of the bus connector.
3. Install the large interface cable between the HSI-24 rear connector and the transducer junction box. Unless you have special cables the ends of the large cable are interchangeable.
4. Replace the computer covers.
5. Start the computer.
6. Make a subdirectory for the HSI-24 software.
7. Copy all the files from the supplied diskette to the HSI-24 subdirectory.
8. Run the batch file called **RUN**.

Files on Diskette

Files Required to run HSI-24

PCA.BIN	firmware which is loaded into coprocessor
PCALDR.EXE	program to load firmware into HSI-24

Files to Run HSI-24 Demo Program

HSIDEMO.EXE	program which displays readings on the host screen.

Files Containing Source of HSI-24 Demo Program

HSIDEMO.C	source of HSIDEMO.EXE
HSIDCMD.C	source of HSIDEMO.EXE
HSIDUTIL.C	source of HSIDEMO.EXE
HSIDCFG.C	source of HSIDEMO.EXE
GETOPT.C	source of command line option scanner
PCALDR.C	source of loader routine
PCACMD.H	HSI-24 command codes
HSIDEMO.MAK	"make" file for hsidemo
HSIDEMO.LNK	linker script for HSIDEMO.EXE
BARS40H.OBJ	object code for 40 bars

Files Not on Diskette

Probe Products Corporation has versions of the HSI-24 communications functions which are for use with BASIC and PASCAL. These files are available free upon request.

Probe Products Corporation has a full featured data collection and SPC display program called EasyMeasure. The EasyMeasure software supports interacive data collection and charting with EGA/VGA graphics. Call for a free demonstration disk and pricing information.

Several commercial suppliers of SPC software have integrated support for the HSI-24 into their software products. Call Probe Products Corporation or your SPC software vendor to inquire about support for the HSI-24.

HSIDEMO Application/Setup Program

General Description

HSIDEMO.EXE is a program which demonstrates many of the features of the HSI-24. It is an easy way to evaluate the functions of the HSI-24 without writing any software. The source code is included as a guide to writers of specialized application software using the HSI-24. The file HSICOMM.C includes functions which are callable from such application software and should be used as a foundation for communication with the HSI-24.

The program HSIDEMO.EXE will construct up to 96 ’channels’ of input-derived data, using from one to four HSI-24 systems. The channels may be viewed eight at a time in tabular form, or up to forty total across the screen of a CGA/EGA/VGA or Hercules monitor in bargraph form. In tabular form the left and right arrow keys shift the channel display window.

A channel is defined using a formula which may be any combination of transducers, analog inputs, functions, arithmetic operators or other channels. The functions provided include MAX HOLD, MIN HOLD, TIR, SINE, COSINE, ARCTANGENT, etc. A channel may be offset by any amount to produce readings that correspond to actual part dimensions in any convenient engineering units, or deviation from zero.

The MAX, MIN, and TIR functions are periodically recomputed. The scanning time for this computation is programmable. The scanning may also be stopped to allow the readings to be frozen.

Both transducer and analog inputs may be assigned an offset and full scale value. The offset is used as a zeroing aid. The full scale value sets the output of the transducers and analogs to the desired engineering units.

Normally a channel value is computed when the HSI-24 receives a request for that channel. Certain functions (MAX, MIN and TIR) are periodically computed.

*PLEASE NOTE: Although the term "LVDT" is used within commands in this document, many other transducers may be used with the HSI-24. Should half-bridge and LVDT transducers be mixed within a system, the sensitivity of all units may be matched for compatibility. Also, many strain gauges, and most potentiometric transducers can be operated with AC excitation, so those types may also be used. Contact PROBE PRODUCTS CORPORATION for suggestions on specific applications.

Data Logging/Printing

There are two methods of causing data to be logged. The **F1** key can be used at any time to log readings or a timer can be set up to periodically store readings at a selectable rate. The radings can be stored to a file, the printer or both.
If the /F was not used during startup and **F1** is pressed, HSIDEMO will store the data to the file named default.dat.. The **F1** key will store the values of each channel which is defined by a formula. The channels are stored in the file in ascending channel number order. Note that HSIDEMO should be terminated with the **Q** command before system shutdown to ensure that the latest data from the buffer is fully written into the output file.

CAL40B.CFG Setup File

The CAL40B.CFG setup file performs the following:
Defines 40 channels with transducer inputs 1 to 24
Sets FSV on all 24 transducers to .040
Sets bar graph scales on all 24 bars to .040
Sets over limit on all 24 bars to .030
Sets high approach on all 24 bars to .020
Sets low approach on all 24 bars to -.020
Sets under limit on all 24 bars to -.030
Sets scan time to 1000. (100 milliseconds)
Enables scanning

HSIDEMO Default Setup

If no configuration file is loaded the defaults are:
No channel definitions
Zero offset on all 96 channels is 0
FSV on all 96 transducers is .020
Zero offset on all 96 transducers is 0
FSV on all 16 analog inputs is 1.00
Zero offset all 16 analog inputs is 0
Scale on all bars is .020
Over limit on all bars is .018
High approach on all bars is .015
Low approach on all bars is -.015
Under limit on all bars is -.018
Scan time is 100 (10 milliseconds)
Scanning enabled

HSIDEMO Startup Options

HSIDEMO has the following command line options:
/a sets an alternate I/O address for the HSI-24
/f sets the filename for both timed and **F1** key file storage.
/s loads the setup file named immediately after the option.

HSIDEMO Menu 1

The following keys are active in Menu 1:

escape	Switch to menu 2.
Q	Stops the program and returns to DOS
R	Resets max, min and run-out readings
C	Save/Load of subsystem configuration
F	Define channel formula
T	Set scanning time
E	Clear all channel formulae
S	Turn scanning on/off
D	Enables display of diagnostic data to/from the HSI-24
Z	Sets bargraph scale
B	Displays bargraphs
L	Sets bargraph limits
I	Sets number of decimal places
P	Printing/File setup for timed measurements

HSIDEMO Menu 2

The following keys are active in Menu 2:

escape	Switch to menu 1.
1	Set zero offset for a transducer.*
2	Auto zero one or all transducers.
3	Remove zero offsets for one or all transducers.
4	Set transducer full scale value.
5	Set zero offset for an analog input.*
6	Auto zero on or alle analog inputs.
7	Remove zero offsets for one or all analog inputs.
8	Set full scale value for an analog input.
9	Set number of readings to average.
0	Set length of running average filter.
M	Set multiplexor settling time. (Used for production test only)
A	Set zero offset for a channel*.
B	Auto zero one or all channels.
C	Remove zero offsets for one or all channels.
R	Resets max, min and runout readings

*NOTE: These functions can be used to display actual sizes rather than deviation-from-zero.

HSIDEMO Channel Scrolling Keys

The following keys are active in both Menu 1 and Menu 2:

down-arrow	Displays next lower channel.
left-arrow	Displays next lower channel.
up-arrow	Displays next higher channel.
right-arrow	Displays next higher channel.
page-down	Displays next lower 8 channels.
page-up	Displays next higher 8 channels.
home	Displays channels 1 to 8.
end	Displays channels 89 to 96.
F1	Stores/prints channel values
escape	Toggles between menu 1 and menu 2

HSIDEMO Bar Graphs

The bar graphs display the current values of up to 40 channels. A CGA monitor will display green, yellow and red bars depending on how the displayed value relates to the limit points, while a Hercules-type monitor will (obviously) display monochrome bars. Each bar has an independent scaling factor, over limit, high approach limit, low approach limit and under limit.

The bars are displayed from menu 1 with the **B** command. To return to menu 1 use the escape key. While the bars are displayed the **F1** key (store data to file), **R** key (reset MAX/MIN/TIR) and **S** key (start/stop scanning) may be used. The **F1** key will not display a warning message if no data output file was defined and **F1** is used while the bars are displayed. No harm is done in this case, except the data will not be stored.

The scaling factor for the bars is set via the menu 1 **Z** command. The scaling factor for each bar graph may be set independently of the other bar graphs. The scaling factor would normally be set to the FSV of the formula which defines the channel for the bar.

Example 1: If Channel 1 = T1 and the FSV of T1 is .04 inches, set the scale for bar 1 to .04.

Example 2: If Channel 1 = T1+T2 the FSV of T1 is .04 inches and the FSV of T2 is .04 inches, set the scale for bar 1 to .08.

Other scaling factors may be used as desired for applications not requiring full transducer stroke.

The limit values for the bars are set via the menu 1 **L** command. The limits for each bar graph may be set independently of the other bar graphs. There are four limits associated with each bar graph; over limit, high approach, low approach and under limit. The over limit and high approach must be zero or greater. The under limit and low approach must be zero or less (negative values). In addition the limit values must satisfy "over limit = high approach = low approach = under limit". If the limit values do not satisfy this relationship the **B** command will display an error message, so if this occurs check all limits to be sure they all satisfy the above relationship. When using the **L** command the present value of the limit is displayed in parentheses. If you do not wish to change the displayed value hit the **Enter** key without typing a new value.

If, on a CGA monitor, you want the bars to display only green and red (no approach limits in yellow), set the over limit and the high approach limit to the same value. This causes the approach region to be skipped. The same idea can be used for the under and low approach limits.

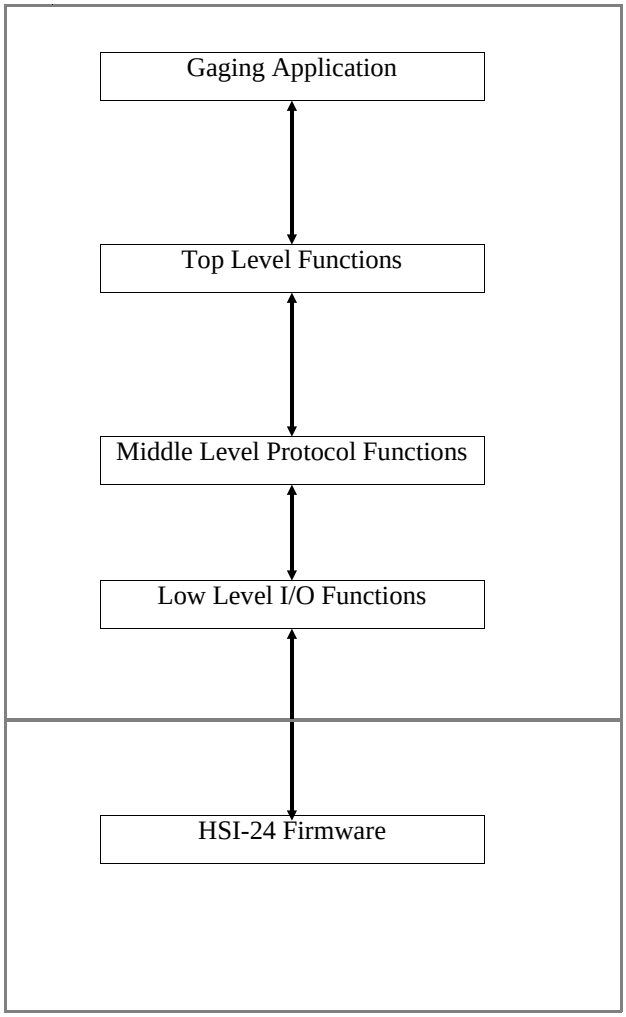
The bar graph display will only display as many bars as there are channels defined. It will never display more than 40 bars due to screen width limitations. The bar graph display will stop at the first channel which has no formula.

Software Interface

Overview

The file HSICOMM.C contains three levels of HSI-24 communication functions. Each level is constructed using components from lower levels. The top (and easiest to use) level implements HSI-24 commands. The middle level implements block I/O functions with the HSI-24. The lowest level implements single character I/O functions with the HSI-24.

To use the HSI-24 it is only necessary to understand the top level functions. The descriptions of the middle and low level functions can be skipped.



Resident in PC

Resident in HSI-24

Top Level Communication Functions

The top level of functions contains one function for each of the HSI-24 commands. To use the functions in the top level, call the function with its arguments. The function will issue the proper command code and arguments to the HSI-24. The function will return the HSI-24's response.

Status Code

All HSI-24 functions return a status code which indicates the success or failure of the command execution. The status code is returned as the value of the function. The functions that return values in addition to the status code require arguments which are a pointer to a place to put the returned value.
The following rules pertain to all top level functions:
All functions return an integer which will be:

zero	If the command was completed successfully.
-1	If the low level routines were unable to communicate.
+value	If the command was not executable. (see status codes)

General Arguments

When functions require a channel number as an argument the channel number must be between 1 and 96.
When functions require an transducer number as an argument the transducer number must be between 1 and 96. (Transducer numbers greater than 24 will be accepted but are only meaningful if slave HSI-24s are installed.)
When functions require an analog number as an argument the analog number must be between 1 and 16. (Analog numbers greater than 4 will be accepted but are only meaningful if slave HSI-24s are installed.)

General Return Values

All functions return the status as the function return value. When other values must be returned the calling routine must supply a pointer to a variable which will have the return value stored into it.

Floating Point Arguments

When a function requires a floating point value, the value supplied is dependent on the condition of the floating point input mode. If the floating point input mode is set to the value FP_IMODE_NATIVE (see file PCACMD.H), then the number must be a single precision number in 8087 format. If the floating point input mode is set to the value FP_IMODE_ASCII then the number must be an ASCII string which represents the value. In the ASCII mode the conversion will terminate when an invalid character is detected. The string 'ABC' will produce a value of '0.0'. Scientific notation is not allowed.

Floating Point Return Values

When a function returns a floating point value, the value returned is dependent on the condition of the floating point output mode. If the floating point output mode is set to the value FP_OMODE_NATIVE, then the number will be a single precision number in 8087 format. If the FLOATING POINT OUTPUT MODE is set to the value FP_OMODE_ASCII, then the number will be an ASCII string which represents the value. The number of decimal places in ASCII output mode is controlled by an HSI-24 command.
Important Note. The functions designed for IEEE floats move the floating point number to the variable pointed to by the argument. The functions designed for ASCII floats only set the passed pointer to point to the result string. The result string still located in the receive buffer. If you want to save the result string you must copy it out of the buffer before another HSI-24 function is called.

Funtion hsicomm_init

int hsicomm_init(unsigned hsi_base)
This function must be the first function called before any of the other functions in HSICOMM.C are used. The call hsicomm_init(0xffff) will set the port address to the default value. The argument can be set to other values, but the address jumpers on the board must be changed to reflect the value which is used. This function will return 0 if the port address is an allowable value.

Function set_diag

int set_diag(int diag)
This function will enable the diagnostic display of communicated data if called with a non-zero argument. It will only enable the diagnostics if HSICOMM.C was compiled with 'HSI-DIAG' TRUE. It does not return any value.

Function start_scan

int start_scan()
This function enables the scanning of formulae which contain MAX, MIN or TIR functions. Returns status.

Function stop_scan

int stop_scan()
This function disables the scanning of formulae which contain MAX, MIN or TIR functions. The functions will be unaffected by changes in their arguments. Returns status.

Function define_channel

int define_channel(int channel, char *formula)
This function will define a formula for a channel. The formula argument is a pointer to a string which contains a valid formula. See the description of formulae for details of allowed formats, etc. Returns status.

Function read_formula

int read_formula(int channel, char **formula)
This function will set the pointer formula to point to the ASCIIZ string which is the formula for the specified channel. Note that this function does not copy the string to a destination string. The formula string is located in the receive buffer. Returns status.

Function clear_channels

int clear_channels()
This function will erase the formulae for all channels. Returns status.

Function clear_channel

int clear_channel(int channel)
This function will erase the formula for one channel. Returns status.

Function read_channel

int read_channel(int channel, float *cvalue)
Reads the value of a channel and places the result in the location pointed to by cvalue. Returns status.

Function set_czero

int set_czero(int channel, float *czero)
This function sets the channel offset for a channel. The channel offset is added to the channel value when the read_channel function is used and when the channel is used in a formula for another channel. Returns status.

Function read_czero

int read_czero(int channel, float *czero)
Reads the current zero offset for a channel. Returns status.

Function reset_mm

int reset_mm()
Resets all MIN, MAX and TIR functions. Returns status.

Function read_lvdt

int read_lvdt(int lvdt, float *tvalue)
Reads the current value of a transducer. Returns status.

Function set_tfsv

int set_tfsv(int lvdt, float *tfsv)
Sets the full scale value for a transducer. Returns status.

Function read_tfsv

int read_tfsv(int lvdt, float *tfsv)
Reads the current full scale value for a transducer. Returns status.

Function set_tzero

int set_tzero(int lvdt, float *tzero)
Sets the zero offset for a transducer. The zero offset is added to the transducer value when the read-transducer function is used and also when the transducer is used in a formula. Returns status.

Function read_tzero

int read_tzero(int lvdt, float *tzero)
Reads the current zero offset for a transducer. Returns status.

Function read_analog

int read_analog(int analog, float *avalue)
Reads the current value of an analog input. Returns status.

Function set_afsv

int set_afsv(int analog, float *afsv)
Sets the full scale value of an analog input. Returns status.

Function read_afsv

int read_afsv(int analog, float *afsv)
Reads the current full scale value for an analog input. Returns status.

Function set_azero

int set_azero(int analog, float *azero)
Sets the zero offset for an analog input. The zero offset is added to the analog input value when the read_analog function is used or when the analog input is used in a formula. Returns status.

Function read_azero

int read_azero(int analog, float *azero)
Reads the current zero offset for an analog input. Returns status.

Function reada_channel

int reada_channel(int channel, char **value)
Same as read_channel except value is returned pointing to an ASCIIZ string. Returns status.

Function seta_czero

int seta_czero(int channel, char *czero)
Same as set_czero function except czero is a pointer to an ASCIIZ string. Returns status.

Function reada_czero

int reada_czero(int channel, char **czero)
Same as read_czero function except czero is returned pointing to an ASCIIZ string. Returns status.

Function reada_lvdt

int reada_lvdt(int lvdt, char **value)
Same as read_lvdt function except value is returned pointing to an ASCIIZ string. Returns status.

Function seta_tfsv

int seta_tfsv(int lvdt, char *tfsv)
Same as set_tfsv function except tfsv is a pointer to an ASCIIZ string. Returns status.

Function reada_tfsv

int reada_tfsv(int lvdt, char **tfsv)
Same as read_tfsv function except tfsv is returned pointing to an ASCIIZ string. Returns status.

Function seta_tzero

int seta_tzero(int lvdt, char *tzero)
Same as set_tzero function except tzero is a pointer to an ASCIIZ string. Returns status.

Function reada_tzero

int reada_tzero(int lvdt, char **tzero)
Same as read_tzero function except tzero is returned pointing to an ASCIIZ string. Returns status.

Function reada_analog

int reada_analog(int analog, char **value)
Same as read_analog function except value is returned pointing to an ASCIIZ string. Returns status.

Function seta_afsv

int seta_afsv(int analog, char *afsv)
Same as set_afsv function except afsv is a pointer to an ASCIIZ string. Returns status.

Function reada_afsv

int reada_afsv(int analog, char **afsv)
Same as read_afsv function except afsv is returned pointing to an ASCIIZ string. Returns status.

Function seta_azero

int seta_azero(int analog, char *azero)
Same as set_azero function except azero is a pointer to an ASCIIZ format floating point. Returns status.

Function reada_azero

int reada_azero(int analog, char **azero)
Same as read_azero function except azero is returned pointing to an ASCIIZ string. Returns status.

Function set_scan_time

int set_scan_time(int time)
Sets scanning period for MAX, MIN and TIR functions. The argument 'time' is in units of tenth's of milliseconds. Returns status.

Function get_scan_time

int get_scan_time(int *time)
Sets time to point to current scanner period. Returns status.

Function set_fp_in

int set_fp_in(int format_code)
Sets floating point input mode. The allowed values for format_code are:
FP_IMODE_NATIVE IEEE single precision floats(8087 format)
FP_IMODE_ASCII ASCIIZ strings
FP_IMODE_MSFP Microsoft fp format (used in Basic)
These values for format_code are defined in file PCACMD.H. Returns status.

Function get_fp_in

int get_fp_in(int *format_code)
Sets format_code to point to current fp input format. Returns status.

Function set_fp_out

int set_fp_out(int format_code)
Sets the floating point output mode. The allowed values for format_code are:
FP_OMODE_NATIVE IEEE single precision floats(8087 format)
FP_OMODE_ASCII ASCIIZ strings
FP_OMODE_MSFP Microsoft fp format (used in Basic)
These values for format_code are defined in file PCACMD.H. Returns status.

Function get_fp_out

int get_fp_out(int *format_code)
Returns format_code pointing to current fp output format. Returns status.

Function set_fp_prec

int set_fp_prec(int precision)
Sets the number of decimal places for floating point numbers when the floating point output mode is ASCII. Numbers are rounded to the specified number of decimal places. The allowed values for the argument precision are between 1 and 6. Returns status.

Function get_fp_prec

int get_fp_prec(int *precision)
Returns precision pointing to current fp precision. Returns status.

Function get_scan_flag

int get_scan_flag(int *scanflag)
Returns scanflag pointing to current scan flag. Returns status.

Function set_mux_time

int set_mux_time(unsigned mtime)
Sets the analog multiplexor delay constant. Returns status..

Function get_mux_time

int get_mux_time(unsigned *mtime)
Returns mtime pointing to current mux settling time. Returns status.

Function read_direct

int read_direct(int lvdt_count,unsigned char *lvdt_list,int **values)
int lvdt_count is the number of lvdt's to be read by this call.
unsigned char *lvdt_list is a pointer to a list of the lvdt numbers which are to be read by this call. The lvdt numbers are 8 bit integers starting at "0" through "23" for the first HSI-24, "24" through "47" for the second HSI-24, etc. The first four DC inputs are "96" through "99", the next four DC inputs are "100" through "103", etc.
int **values is a pointer to an integer pointer. The function will set the supplied integer pointer to address the list of returned integers. Each of the returned integers will be in the range of -8192 to +8191.
The function returns status.

Middle Level Functions

The middle level consistes of two functions: send_command and get_response.
All communication with the HSI-24 is done through the writing and reading of blocks of data. Communication is always initiated from the host (IBM-PC) side. The slave (HSI-24) always responds with a command completion status.

Blocks to HSI-24

The PC to HSI-24 blocks have the following structure:

byte 1 3Ah (ASCII ':') -- this is the 'host prefix character'.

byte 2 command number -- the command that the HSI-24 is to execute.

byte 3 data count -- count of the number of data bytes which follow. A count of 0 is interpreted as a count of 256. Up to 256 data bytes follow.

byte 4 data -- The first data byte. Note that at least one data byte must be sent.

byte N data -- last data byte.

Blocks from HSI-24

The HSI-24 to PC blocks have the following structure :

byte 1 3Bh (ASCII ';') -- this is the 'slave prefix character'.

byte 2 status -- the command completion status.

byte 3 data count -- count of the number of data bytes which follow. A count of 0 is interpreted as a count of 256. Up to 256 data bytes follow.

byte 4 data -- The first data byte. Note that at least one data byte will be sent.

byte N data -- last data byte.

The HSI-24 will always respond with a block that has as a minimum 'prefix', 'status', count of 01, dummy data byte 00.

After a block is written to the HSI-24 the PC must wait for the response from the HSI-24. When the HSI-24 is ready to respond it will set the 'SREQ' flag bit. The PC resets the 'SREQ' flag and reads a block from the HSI-24.

Function send_command

int send_command(char command, int count, char *data_buffer)
char command - This is the command code to the HSI-24. The code must be one of the codes defined in the file 'PCACMD.H'. Any other code will cause a returned status meaning invalid command.
int count - This is the number of characters to be sent to the HSI-24. When sending strings to the HSI-24 (such as formulae) the count must include the 0(NULL) which terminates the string.
char *data_buffer - This is a pointer to the data that will be sent to the HSI-24.

Function get_response

int get_response(int *psize, char *data_buffer)
int *psize - This is a pointer to an integer variable which will be filled in with the number of characters received from the HSI-24. This count does not include the 'slave prefix' or the 'status code'.
char *data_buffer - This is a pointer to a buffer which will be filled with the data received from the HSI-24.
Return value from get_response is the 'status code' from the HSI-24. This code indicates the success (or failure) of the command.

HSI-24 Timeout

Both send_command and get_response functions will return a -1(0xffff) if the lower level communication functions cannot communicate with the HSI-24 in a 'reasonable length of time'. The length of a reasonable length of time is set by the '#define CMD_TIME' statement.

Low Level Functions

The three low level functions are responsible for the actual transfer of data between the PC and the HSI-24.

Function pbyte

int pbyte(unsigned char data)
Writes the argument data byte to the HSI-24. Returns 0 if successful. Returns -1 (0xffff) if HSI-24 does not indicate 'ready for data' within a reasonable amount of time.

Function gbyte

int gbyte()
Reads one data byte from the HSI-24. Returns the data byte read, if successful. Returns -1 (0xffff) if HSI-24 does not indicate 'data ready' within a reasonable amount of time.

Function xfer

int xfer()
The xfer function waits for an HSI-24 command to complete. Returns 0 if successful. Returns -1 (0xffff) if HSI-24 does not indicate 'complete' within a reasonable amount of time.

Customization of hsicomm.c

The file HSICOMM.C contains several #define variables which can be used to customize some of the characteristics of the functions produced.

FP_NATIVE - Set to TRUE if you are using 8087 format fp numbers.

FP_ASCII - Set to TRUE if you are using ASCII format fp numbers.

CMD_TIME - This sets the length of time the low level I/O routines will wait before reporting a time-out error. The HSI-24 will respond to commands within several milliseconds (for a channel with a complex formula) and several hundred microseconds for a simple formula. The response time is about 400 microseconds per operation.

HSI_DIAG - Set TRUE to include code which will display communicate data.

ROW_FOR_DIAG - CRT row where diagnostic data is displayed.

COL_FOR_DIAG - CRT column where diagnostic data is displayed.

HSI_STAT - Set TRUE to include code which will display non-zero status bytes.

ROW_FOR_STAT - CRT row where status will be displayed.

COL_FOR_STAT - CRT column where status will be displayed.

(Note: the diagnostic and status display code use a cursor locating routine which you must supply if you use these options.)

HSI-24 Formula Syntax

When a channel is read, its value is computed by a formula supplied to the HSI-24 by the host software. The formulae are supplied to the HSI-24 in the form of an ASCII string terminated by a null byte. Formulae are constructed in much the same way as they are in assignment statements in programming languages such as BASIC.

Formulae consist of combinations of functions, operators, constants, input terms and channel terms.

Peak Hold Functions

MAX returns the largest value it sees as an argument while scanning.
MIN returns the smallest value it sees as an argument while scanning.
TIR returns MAX-MIN.
The MAX, MIN and TIR functions continue to be scanned although the channel which uses them is not being read by the host software. They will 'freeze' at their current value if scanning is disabled. The scanning is controlled by the functions start_scan and stop_scan. The period of scanning is set by the set_scan_time function. They are reset by the reset_mm function.

Mathematical Functions

ABS returns the absolute value of the argument.
ACOS returns the arc-cosine of the argument. Result in radians.
ASIN returns the arc-sine of the argument. Result in radians.
ATAN returns the arctangent of the argument. Result in radians.
COS returns the cosine of the argument. Argument in radians.
GOF returns the greatest (most positive) of the argument list.
GOR returns the greatest (most positive) of the range set by the argument list.
LOF returns the least (most negative) of the argument list.
LOR returns the least (most negative) of the range set by the argument list.
SIN returns the sine of the argument. Argument in radians.
SQRT returns the square root of the argument.
SQR returns the argument multiplied by itself.
TAN returns the tangent of the argument. Argument in radians.

Miscellaneous Functions

PI2 has the value 3.141592654/2.0
PI has the value 3.141592654
RAD returns the argument converted from degrees to radians.
DEG returns the argument converted from radians to degrees.
(and **)** are allowed to impose computaion order.

Mathematical Operators

^ exponeniation
***** multiplication.
/ division. (division by zero returns zero
+ addition.
- subtraction.

Constants

Constants in formulae are specified as ASCII strings. Scientific notation is not allowed for constants, (i.e., '.125' is allowed, '1.25E-01' is not allowed).

Input Terms

Tn returns the value of transducer number n. The number n must be in the range 1 to 96. The HSI-24 will read the transducer, multiply by the transducer full scale value and add the transducer offset. Each transducer has an individual full scale value and offset.
An causes the value of analog number n to be returned. The number n must be in the range 1 to 16. The HSI-24 will read the analog input, multiply by the analog full scale value and add the analog offset. Each analog input has an individual full scale value and offset.

Channel Terms

Cn returns the value of channel number n. The number n must be in the range 1 to 96. The HSI-24 will compute the value of channel nn, multiply by the channel full scale value and add the channel offset. Each channel has an individual full scale value and offset. Circular references to channels are not allowed and will cause the HSI-24 to return an error code.

Example Formulae

T1
T1+T2
MAX(T2-T1)
(T1+T2+T3)/3
1.0034*(T1+T2)
(MAX(T1)+MIN(T1))/2
GOF(T1,T2,T3) returns the greatest of T1, T2 and T3.
LOF(T1,T2,T3,T4) returns the least of T1, T2, T3 and T4.
GOR(T1,T8) returns the greatest of T1,T2,T3,T4,T5,T6,T7 and T8.
LOR(C9,C13) returns the least of C9,C10,C11,C12 and C13.

HSI-24 Command Codes

Below are listed all of the mneumonics and their values for the HSI-24 commands. This information was derived from the contents of the file PCACMD.H. Although none of the values will change, additional command codes may have been added. Check the contents of the PCACMD.H file included with the HSI-24 to see if any new commands have been added.

C_NOP	0	Dummy command.
C_READ_MEM	1	Read 80186 memory block.
C_DIRECT_ADC	2	Read A/D directly.
C_CHAN_DEFINE	3	Define channel formula.
C_CHAN_READ	4	Read channel.
C_CHAN_CLR_ALL	5	Erase all channel formulae.
C_CHAN_CLR	6	Erase specified channel formula.
C_CHAN_SCAN_ON	7	Start scanning.
C_CHAN_SCAN_OFF	8	Stop scanning.
C_CHAN_RESET_MM	9	Reset max/min nodes.
C_SET_SCAN_TIME	10	Set scanner period.
C_LVDT_READ	11	Read lvdt transducer.
C_ANALOG_READ	12	Read analog transducer.
C_CHAN_FORM	13	Read back channel formula.
C_TFSV_SET	21	Set probe full scale values.
C_TFSV_READ	22	Read probe full scale values.
C_AFSV_SET	23	Set analog full scale values.
C_AFSV_READ	24	Read analog full scale values.
C_TZERO_SET	32	Set probe offset values.
C_TZERO_READ	33	Read probe offset values.
C_AZERO_SET	42	Set analog offset values.
C_AZERO_READ	43	Read analog offset values.
C_CZERO_SET	52	Set channel offset value.
C_CZERO_READ	53	Read channel offset value.
C_CSCALE_SET	54	Set channel scaling value.
C_CSCALE_READ	55	Read channel scaling value.
C_SET_FP_IMODE	80	Select fp input format.
Allowed modes:		
FP_OMODE_NATIVE	0	IEEE single precision floats.
FP_OMODE_ASCII	1	ASCII strings.
FP_OMODE_MSFP	2	Microsoft fp format (used in Basic).
C_SET_FP_OMODE	81	Select fp output format.
Allowed modes:		
FP_IMODE_NATIVE	0	IEEE single precision floats.
FP_IMODE_ASCII	1	ASCII strings.
FP_IMODE_MSFP	2	Microsoft fp format (used in Basic).
C_SET_MUX_TIME	82	Set mux settling time.
C_SET_FP_PREC	83	Set fp precision.
C_GET_FP_IMODE	84	Return current fp input format.
C_GET_FP_OMODE	85	Return current fp output format.

C_GET_MUX_TIME	86	Return current mux settling time.
C_GET_FP_PREC	87	Return current fp precision.
C_GET_SCAN_FLAG	88	Return scan flag.
C_GET_SCAN_TIME	89	Return scanner period.
C_SET_ADC_AVG	90	Set adc averaging quantity.

Startup Settings

Immediately after loading the HSI-24 firmware the following settings will be valid.
All 96 channel scale factors are set to 1.
All 96 channel zero offsets are set to zero.
All 96 transducer full scale values are set to .08.
All 96 transcducer zero offsets are set to zero.
The first 4 analog full scale values are set to 1.
The remaining 12 analog full scale values are set to zero. (Although these are accessible they are physically located on slave boards.
All 16 analog zero offsets are set to zero.
The floating point input and output modes are both set to the native (IEEE) mode.

Error Codes

0 Normal return which means the command was acceptable.

1 An invalid command or parameter was entered.

2 thru 9 are reserved.

When defining channels the following error codes may be returned. They may be used to help determine the part of the formula which is causing the problem.

10 An invalid channel number was entered.

11 Internal error..

12 Invalid opcode mneumonic. A built-in function name has been improperly typed in. If "SINE(T4)" is entered instead of "SIN(T4)" or "T1+S3" instead of "T1+T3".

13 Not enough operands for operator. If "T1+" is entered the HSI-24 cannot determine what value to add to T1. If "SIN()" is entered the HSI-24 cannot determine what value to use as the argument of the sine function.

14 Node table full. The HSI-24 reduces the entered formulae into a form which can be quickly computed when required. The reduced formulae are stored in a table which has a predetermined size. When the table is filled this error is returned. If this error occurs, the number or the complexity of the entered formulae must be reduced.

Determining the number of node entries in the node table that a particular formula consumes is difficult due to certain optimizations which the HSI-24 applies during the formula reduction process. The following guidelines will allow determining the maximum number of nodes which a given formula will consume.
1. Each constant consumes one node. The formula ".0023" consumes one node.
2. Each function use consumes one node for the function. Additional nodes will be consumed for the function's arguments. The formula "SIN(.0023)" consumes one node for the SIN function plus one node (Rule 1) for the constant .0023, for a total of 2 nodes. The TIR function is a special case which consumes 2 nodes just for the function plus the additional nodes for the function arguments. The formula "TIR(.0023)" consumes 2 nodes for the TIR function and 1 node for the constant, for a total of 3 nodes.
3. Arithmetic operators (+ - * /) consume one node. The formula "1 + 2 + 3" consumes 1 node for each of the "+" operators plus 1 node for each of the constants (Rule 1), for a total of 5 nodes.
4. Channel references consume one node. The formula "C1" consumes 1 node.
5. Transducers and analog inputs consume one node. The formula "T1" consumes 1 node. The formula "T1 + T2" consumes 1 node for each of the transducers plus 1 node for the "+" operator (Rule 3), for a total of 3 nodes. The first appearance of a transducer in a formula consumes 1 node however subsequent appearances of the same transducer in any formula do not consume additional nodes.
Channel 1 is "T1+T2". This uses 3 nodes.
Channel 2 is "T1+T3". This uses 2 nodes. T1 has already been allocated a node by the channel 1 formula.

The total number of nodes available is 400 in firmware version ’PCA10.BIN’.
The HSI-24 cannot conserve nodes by recognising common subexpressions in formulae. Node table space can be conserved by defining a channel with the subexpression and then referencing that channel in the formulae which need the subexpression value.

Channel 1 is "T5 - (T1 + T2 + T3 + T4)". (9 nodes)
Channel 2 is "T6 - (T1 + T2 + T3 + T4)". (5 nodes)
Channel 3 is "T7 - (T1 + T2 + T3 + T4)". (5 nodes)
Channel 4 is "T8 - (T1 + T2 + T3 + T4)". (5 nodes)
For a total of 9+5+5+5 or 24 nodes.

Alternately define channel 50 with the subexpression and use channel 50 in place of the subexpression:
Channel 50 is "T1 + T2 + T3 + T4". (7 nodes)
Channel 1 is "T5 - C50". (3 nodes)
Channel 2 is "T6 - C50". (3 nodes)
Channel 3 is "T7 - C50". (3 nodes)
Channel 4 is "T8 - C50". (3 nodes)
For a total of 7+3+3+3 or 16 nodes.

15 Bad transducer or analog input number. If a formula contains "T175" or "A32". The highest transducer number allowed is "T96". The highest analog input number allowed is "A16". Note: Although the HSI-24 accepts transducer numbers up to 96 the slave HSI-24’s must be present for these to produce any meaningful measurements.
16 Too many operands for operator. If "T1 + T2 T3" is entered, the HSI-24 cannot determine what to do with the T3 part of the formula.
17 Bad numeric value in expression. When entering constants in a formula "scientific notation" is not allowed. In some computer languages the value .0015 may be entered as "1.5E-3". The HSI-24 requires that the value be entered as ".0015".
18 Bad token. An invalid symbol was entered.
19 Formula too complex to parse. This error occurs when a formula has more levels of parenthesis than the HSI-24 can handle
20 Recursive channel definition. This error occurs when channels reference each other in a circular fashion.
Channel 1 has the formula "T1+T2-C2".
Channel 2 has the formula "T3+C1".
To determine the value of channel 1 the HSI-24 must first determine the value of channel 2, which is dependent on the value of channel 1.
21 No memory left to allocate. Besides storing the formulae in a reduced form (as described under error 14), the original text of the formula is also saved within the HSI-24 memory. The formula text is stored in a memory pool along with other items which are necessary during formula entry. Should this pool of memory become filled this error is issued.
22 General formula error. The formula scanner in the HSI-24 will issue this error when the formula is bad and no other formula error codes apply.

HSI-24 HARDWARE

The HSI-24 coprocessing subsystem has four major components:

1 to 4 -SYSTEM BOARD containing all active circuitry. This board resides within the host computer.

1 to 4 -CABLE ASSEMBLY for connecting the system board to the junction box. This is a 100 conductor cable assembly--please exercise caution when connecting, to avoid damaging the small pins in the connectors.

1 to 4 -JUNCTION BOX, with 25 DIN connectors, 24 of which are for transducers and one for the four +/-5V DC inputs.

1 -SOFTWARE DISKETTE.

Changing the I/O Address

The HSI-24 subsystem uses four I/O channel addresses on the PC I/O channel. As shipped, the subsystem uses addresses 0x0310 through 0x0313. IBM has assigned these addresses to what they call the "prototype board". In the event of an address conflict with another adapter, the address of the HSI-24 can be changed to one of four preset addresses. There are two pairs of jumper pads located on the back of the HSI-24/14 board near the PC bus connector. The jumpers are labelled JP1A and JP1B. Each jumper pair has a small trace running between its pads. This jumper must be cut to open the jumper. If the trace is already cut a small wire must be soldered across the pads to close the jumper. The table below shows the I/O address selected for all four combinations of jumpers.

J1A	J1B	I/O Address
Closed	Closed	0310 Hex
Closed	Open	0308 Hex
Open	Closed	0318 Hex
Open	Open	02A0 Hex

It is unlikely that you will have to change the I/O address, but it may be necessary if you have an IEEE-488 adapter, more than 4 communication adapters, other A/D converter boards or parallel I/O adapters (other than printer adapters).

Connector Pinout

J2 (transducer Connector) pinout as viewed from rear of PC

		top left	top right	
DC INPUT 1		1	51	DC INPUT 2
T24	+OSC	2	52	T24 SIGNAL
T24	-OSC	3	53	COMMON
T22	+OSC	4	54	T22 SIGNAL
T22	-OSC	5	55	COMMON
T20	+OSC	6	56	T20 SIGNAL
T20	-OSC	7	57	COMMON
T12	-OSC	8	58	T2 SIGNAL
T12	+OSC	9	59	COMMON
T10	-OSC	10	60	T4 SIGNAL
T10	+OSC	11	61	COMMON
T8	-OSC	12	62	T6 SIGNAL
T8	+OSC	13	63	COMMON
T6	-OSC	14	64	T8 SIGNAL
T6	+OSC	15	65	COMMON
T3	-OSC	16	66	T10 SIGNAL
T3	+OSC	17	67	COMMON
T2	-OSC	18	68	T12 SIGNAL
T2	+OSC	19	69	COMMON
T14	+OSC	20	70	T14 SIGNAL
T14	-OSC	21	71	COMMON
T16	+OSC	22	72	T16 SIGNAL
T16	-OSC	23	73	COMMON
T18	+OSC	24	74	T18 SIGNAL
T18	-OSC	25	75	COMMON
T23	+OSC	26	76	T23 SIGNAL
T23	-OSC	27	77	COMMON
T21	+OSC	28	78	T21 SIGNAL
T21	-OSC	29	79	COMMON
T19	+OSC	30	80	T19 SIGNAL
T19	-OSC	31	81	COMMON
T11	-OSC	32	82	T1 SIGNAL
T11	+OSC	33	83	COMMON
T9	-OSC	34	84	T3 SIGNAL
T9	+OSC	35	85	COMMON
T7	-OSC	36	86	T5 SIGNAL
T7	+OSC	37	87	COMMON
T5	-OSC	38	88	T7 SIGNAL
T5	+OSC	39	89	COMMON
T3	-OSC	40	90	T9 SIGNAL
T3	+OSC	41	91	COMMON
T1	-OSC	42	92	T11 SIGNAL
T1	+OSC	43	93	COMMON
T13	+OSC	44	94	T13 SIGNAL
T13	-OSC	45	95	COMMON
T15	+OSC	46	96	T15 SIGNAL
T15	-OSC	47	97	COMMON
T17	+OSC	48	98	T17 SIGNAL
T17	-OSC	49	99	COMMON
DC INPUT 4		50	100	DC INPUT 3
		bottom left	bottom right	

The +OSC and -OSC designations refer to the phase of the oscillator drive signal relative to the inward-from-null signal from the transducer.

Adjustments

The trimmer pots at the top of the HSI-24 adjust the gain of each transducer signal conditioner independently. The trimmer nearest the rear connector is for channel 1, the next one is for channel 2, etc. These trimmers allow for adjustment during manufacturing to correct for part value variations, and may also be field adjusted if necessary.

The trimmer below the heatsinks and next to the large group of brown capacitors is the phase adjust pot. With the installation of a capacitor this compensates for the phase shift produced by some transducers. As delivered the phase shift is adjusted to zero.

The trimmer at the middle center of the HSI-24/14 adjust the A/D converter gain. This should not be changed from factory settings.

The trimmer below the heatsinks near the edge of the HSI-24 sets the pre-regulator voltage and should not be changed from factory setting.