Nist Software Users Manual

V3.0

TABLE of CONTENTS

NIST Database Software Users Manual V3.0

by Dave Morrish

1 PURPOSE

The data set accumulated during an extensive set of wind tunnel tests of generic buildings in typical exposures will be used by a large group of researchers for an extended period of time. In the past, the data files consisting of raw time series data files written in a hexadecimal format (for compactness and cross platform readability) have been provided.

Although the above method provided a usable way of conveying the data, details about test parameters, model and wind tunnel configuration were stored separately from the time series files. It became clear that as the number of data sets and configurations increased, the possibility of introducing errors due to miscommunication of experimental parameters became large. The following document details the methods and software proposed to provide a solution to this problem.

The overall objective is to provide the user with all the information needed to define the experiment within which the data were taken, and to allow the user to extract the data needed to form it into a matrix (pressure coefficients for tap i versus time step j) for further analysis. The information provided for the user can include pictures of the model and setup, and whatever ancillary supporting information is deemed necessary to define the tests.

2 CONCEPTS

There are several "standard" data formats used within the scientific community such as:

- CDF defined by NASA which is used by the space physics community.
- Net-CDF used by the meteorological community supported by UCAR (University Corporation for Atmospheric Research).
- HDF format used by NCSA (National Center for Supercomputing Applications).

The data storage formats mentioned above (although not interchangeable) are all governed by the same underlying philosophy. Specifically:

- To isolate the programmer from internal details of the file.
- To provide named self-describing data items. Specifically, array size, dimensions, data type and a descriptive tag (or title) are embedded in each data item.
- To provide machine independent data storage. Various computer manufacturers use different internal formats to represent numeric values. When transferring data files between dissimilar computers data must be converted or a common data storage format must be used.

3 SOLUTION USED FOR BLWTL SUPPLIED FILES

The solution chosen for transmitting the pressure data files is a much enhanced version of our present internal pressure file format. Our internal format consists of a fixed size descriptive header and a large body of binary time series data. The existing format was judged unsuitable for external use due to the complex ordering of the raw data as received from the pressure scanner A/D converter.

An initial format which embodied many of the concepts described in item two, was created at BLWTL and released with sample programs and data files in the May of 2001. Although the BLWTL file format performed well, the amount of effort required for support for multiple languages and computer platforms became a limiting factor. In the spring of 2001 release 12 of Matlab appeared with added support for the HDF r4.1 Format. Based on these two factors it was decided to use the NCSA HDF format for the production release of the data.

In June of 2001 a meeting of researchers involved in the Electronic Pressure Database Project was held. The group discussion yielded many useful suggestions for incorporation into the database. The key suggestions and implementations are as follows:

1. <u>Descriptive File Naming Convention</u>

A file naming scheme which conveys information about major test parameters by encoding them into specific fields of the filename has been adopted.

Typical file name: ADW100o100S048axxxx.HDF

pos 1-3 identifies the originating facility:

Example BLWTL files start with ADW for Alan G. Davenport Wind Engineering Group

pos 4-6 identifies the roof slope in 12ths to 3 digits

Example: 100 is 1 in 12

Example: 300 is 3 in 12

Example: 025 is 1/4 in 12

pos 7 identifies the exposure

Example: o lower case O for open country

Example: s lower case S for suburban exposure

Other exposures should be documented in the "readme" data item Read_Me.

(The Read_Me item is a collection of freeform text paragraphs which can be used to provide additional clarification of any aspect of the test procedure)

pos 8-10 identifies the model scale to 3 digits

Example 100 is a 1 in 100 scale model test.

pos 11 identifies the leakage case

Example D for distributed leakage.

Example S for small opening.

Example L for large opening.

Example N for no leakage.

Example B for basic test (internal pressures not considered).

Other Leakage cases should be identified in the “readme” data item.

pos 12-14 is the building eave height in full scale feet to 3 digits

Example 048 is 48 feet.

pos 15 is the case identifier which is used to differentiate ambiguities where 2 model tests would generates the same file name. This character position contains a lower case letter typically “a”.

pos 16-19 is the wind angle in degrees to be read as xxx.x

Example 1800 is 180.0 degrees

Angle convention should be defined in the “readme” data item.

.HDF identifies the data format as a Hierarchical Data File

which is a standard file format for the exchange of scientific data.

The HDF file format was designed by the National Center for Super-

Computing Applications (NCSA) .

2. Better Data Structures to Support Tap Identification and Locating

The HDF files contain a data item called “Tap_Position_List” which is a vector with the tap numbers in the order that they appear in the data item “Time_Series”. The Time Series data can be thought of as a matrix. Each column is a specific pressure tap and each row is a specific instant in time.

Sample: Tap_Position_List

101 102 103 104 105 106 107 108

* Note tap numbers define scanner and position i.e. Tap 108 is the 8th input on scanner module number 1.

The Data Item **Tap_Coordinates_3D** contains five colums:

1) Column 1 contains the tap number

2) Column 2 contains the face number that the tap is located on. We have enclosed a drawing file (.jpeg) in the image_2 data field which shows the faces referred to by column 2 .

3) Column 3-5 contain the XYZ coordinates for the taps in 3D space dimensions are in full scale feet.

To compliment the tap coordinate information, the data items Building_Corners_3D and Wire_ Frame_Lines_3D can be used to draw a wire frame outline of the building for illustration.

The Data Item **Flat_Tap_Coordinates** the tap locations in a flattened 2D coordinate system which is easier to use for graphing data.

1) Column 1 contains the tap number.

2) Column 2 contains the face number that the tap is located on. We have enclosed a drawing file (.jpeg) in the image_2 data field which shows the faces referred to by column 2 .

3) Column 3-4 contain the XY coordinates for the taps in 2D space dimensions in full scale feet.

To compliment the tap coordinate information the data items Building_Corners_Flat and Flat_Wire_ Frame_Lines can be used to draw a wire frame outline of the building for illustration.

3. Ability to Efficiently Extract an Arbitrary Subset of Taps and Time Steps.

The data handling routines supplied by the NCSA as part of the HDF standard have the ability to extract any slice of any data item within an HDF file. These routines have been incorporated into the Matlab HDF support facility and are used by the function "load_hdf_file.m" which is located in the /matlab folder of the distribution. Load_hdf_file forms the basis of the demonstration program hdf_read_demo.m and hdf_read_demo.exe . Load_hdf_file when called from Matlab will load all of the supporting data structures plus any slice (consecutive list of taps or time steps) into a Matlab structure.

4. Tap Coordinates and Wire frame Layout also Provided in Flat 2D Space

Although not necessary to analyze the data, this optional information makes many types of graphical representations easier. We have provided them where possible. They consist of tap coordinates , building corners and wire frame segments to generate a flat picture of all building faces. The following data items are used to generate flat graphics Building_Corners_Flat , Flat_Wire_Frame_Lines and Flat_Tap_Coordinates. Note: As with the 3D coordinate data structures, drawing line segments from the coordinates of corner pairs specified by wire frame lines will form a "wire frame" illustration of the building.

4 DATA GENERATED AT OTHER FACILITIES

We have included the Matlab source code for nist_db_hdf3 which is the program used at BLWTL to create the HDF files for this project. This program requires several input files. The names of the required input files are assembled based on rules and information contained in control files which act as lookup tables. Although probably not directly usable at other facilities this program could be used as a basis for writing programs to create HDF files.

Nist_db_hdf3 uses a function named put_head_item_hdf.m (provided) which adds the data items to the HDF file. Put_head_item_hdf.m calls the Matlab interface to the HDF routines. The Matlab HDF routines are documented in “Using Matlab version 6” reference manual starting at page 6-29.

5 OVERVIEW OF SOFTWARE

The software supplied consists of MATLAB functions to extract and display data items within an HDF file. The Matlab functions (described in section 7 below) are located in the folder /hdf_read_demo_exe. The main program “hdf_read_demo.m” can be run from within Matlab or compiled into a Windows executable (.exe) program using the script function “build_hdf_read_demo.m” (only if the Matlab compiler is installed on host system). Other Matlab functions such as “load_hdf_file” can be run from within the Matlab environment or incorporated into other Matlab programs or functions. To allow Matlab to find these functions the “addpath” statement can be used.

The Matlab functions used to create the HDF filed are supplied in the /HDF_create_matlab folder.

6 INSTALLATION

The programs, functions and test data are each supplied in separate folders, within the PKZIP file: nist2.zip

6.1 To Install:

1) Create a folder for the installation . In the example below “c:\projects\nist30” was chosen.

2) Copy all the folders on the CD to the directory created in 6.1

3) Add the following to your search path. (for our example c:\projects\nist30\bin\win32) by placing an additional path statement in your autoexec.bat on Windows 98 based systems.

   example: PATH=%PATH%;c:\projects\nist30\bin\win32;

   On Windows 2000 and XP systems: click on : My Computer > Control Panel > System > Advanced > > Environment Variables > System Variables > Click on path and add ;c:\projects\nist30\bin\win32 to the existing search path. NOTE: This may require administrator privileges and be careful not to alter any of the existing text !

4) Reboot your system.

5) Open the Windows Explorer and move to the c:\projects\nist30\bin\win32 folder. Double click on the hdf_read_demo_exe executable. This will open a command prompt window and a input file selection window will pop up.

6) Select a .hdf file from /projects/nist30/test_data . Enter type of output and time history info when prompted. The output files will have the same root name as the input file, extensions will describe the contents of the file. Example **.hed** is a listing of the name and size of the supporting data structures in the HDF file and optionally the contents. .jpg are the JPEG images of the of the model. .BIN .HEX and .ASC are the extensions given to the output time series written in binary (integer*2) , Hexadecimal is written as x4.4 with no spaces and ASCII written f7.3,1x formatting.

Note 1: When reading the HEX values 32768 must be subtracted from the data and then multiply by .001 (Ts_Multiplier) to get pressure coefficients.

Note 2: When reading binary data the values must be multiplied by Ts_Multiplier (usually .001 for BLWTL supplied files) to get pressure coefficients.

Ts_Multiplier is a data item located in the HDF file.

6.2 Path/Folder Structure

\bin (\win32) -contains the windows executables ".exe" files and the required DLL libraries for hdf_read_demo_win.exe

\docs -contains any readme files and PDF format of documentation.

\HDF_create_matlab contains the source code used to read BLWTL internal files and produce HDF files. Provided for example only.

\hdf_read_demo_exe_win contains the Matlab sources and command files necessary to build the windows executable version of hdf_read_demo_win. These functions can also be run from within the Matlab environment. The best way to run this to use the Matlab "addpath" statement to add this directory to the Matlab search path and then use the "cd" command to make the \test_data directory the "working" directory.

\test_data contains a sample .HDF data file. A small set of HDF files have been included. Output time histories and files ".hed and .JPG" are also place in this directory.

\unsupported_utilities contains utility programs and Matlab supplied by NCSA. For support see NCSA HDF web site.

\toolbox contains additional Matlab support files for the executables. Don't Touch**!**

7 MATLAB FUNCTIONS PROVIDED

7.1 LOAD_HDF_FILE

This function loads the contents of all or a selected portion of the time series data into a Matlab data structure. The function is also used in Hdf_read_demo which can be run as a standalone executable program (hdf_read_demo.exe) or from within the Matlab environment hdf_read_demo.m provided. To use these functions use the Matlab "addpath" statement to add the folder where these routines are stored to your Matlab search path. Example

Used Like This:

Pfile_struct=load_hdf_file(hdf_file_name,load_ts_flag,st,et,si,ei)

or

pfile_struct=load_hdf_file(hdf_file_name,load_ts_flag)

-hdf_file_name String containing full name and path of the HDF file.

-load_ts_flag 'a' to load entire time history all taps

'n' to load NO time history , only the supporting info is loaded.

's' range of taps and samples

If the s option is chosen four additional arguments are required.

-st starting tap # example 1 101 -1 defaults to min. tap index

-et ending tap # example 6316 -1 defaults to max. tap index

-si starting index 1 > No_Of_Samples_Per_Tap -1 defaults to min. sample index

-ei ending index 1 > No_Of_Samples_Per_Tap -1 defaults to maximum sample index

NOTE: calling load_hdf_file with 's',-1,-1,-1,-1 should produce the same

results as using the 'a' option

Returns a structure containing all the supporting information with

or without the time series based on user input . Specifically the load_ts_flag

and st et si ei arguments. Names of attributes appear before the data set that they describe and begin with "A_".

Note: The time history is returned as integer*2 (2 bytes) to conserve memory.

To convert the time history data to actual units (Cp's), the data must be

multiplied by "structure name ".Ts_Multiplier.

Examples:

x=load_hdf_file('ADW100o100L0162000.HDF','a') loads all of the HDF time history info into the Matlab Structure "x" . The time series data would be accessed by x.Time_Series and would be stored as a matrix. The rows would be time steps and the colums would be individual taps. The tap numbers / column titles would be described by x.Tap_Position_List

y=load_hdf_file('ADW100o100L0162000.HDF','s',101,108,1,1000) loads time series data from the first eight taps 108-108 for the first 1000 measurements in Matlab structure "y". The time series data would be accessed by y.Time_Series

For more details on the use of structures within Matlab see "Using Matlab version 6" by Mathworks Chapter 20

7.2 Print_hdf_struct

This function takes a Matlab data structure produced by load_hdf_file and writes it out to files based on the input arguments supplied.

Used like This:

print_hdf_struct(struct,mode_flag,output_file_name,ts_write_flag)

Struct: Matlab file structure containing the contents of an HDF file produced by calling load_hdf_file

Mode_flag: 's' for summary; 'c' for contents. This controls the contents of the .hed file 's' for summary of the size and type of data . A 'c' in this argument will output the contents of all of the supporting data items to the .hed file. The 'c' option also causes the images to be written to .jpg files.

output_file_name: path and file name to write the information to .

Note : does not output contents of struc.Time_Series to output_file_name

ts_write_flag: 'n'= no 'a'=ASCII 'b'=binary 'h' for hex

Creates a separate output file output_file_name.asc or .bin for binary or .hex for hexadecimal.

Caution: specifying ASCII for all taps and all scans can result in huge files and will

take a long time to run.

Examples:

Print_hdf_struct(x,'c','xout.hed','a') Would print all of the supporting information for this file (excluding images) into a file xout.hed in a format similar to Appendix A of this document. The images would be exported as jpeg files to xout_image1.jpg up to xout_image_3.jpg (only the first image is mandatory). The time history would be exported as a column bases ASCII text file formatted as F7.3, 1x. to xout.asc

7.3 Other Functions

For details of other utility functions found in this directory type "help filename" from the Matlab prompt where filename is the name of the Matlab function.

Appendix A Data Structures within an HDF file

> File_Name [1 23] Type: char

ADW025o100D012a0600.HDF

> Title [1 70] Type: char

nist3;open c;12ft bld;60deg

> BLWTL_NIST_Software_Version [1 1] Type: double

3.000000

> READ_ME [1 4648] Type: char

**** AGD Wind Engineering Group ****

NIST Data Storage

This document provides general information for the accompanying

AGD Wind Tunnel raw data and Matlab programs

Data Collection:

The wind tunnel time series were collected over a period of 2 years in AGD Wind Tunnel II. See the following report for basic information on the test parameters used in the wind tunnel tests at the AGD Wind Engineering Group.

Ho, T.C.E., Surry, D. Wind tunnel testing of low buildings, The University of Western Ontario, BLWT-SS11-2000, 2000.

****

File Naming Convention:

The low-rise building data-set contains raw data and statistical information for a number of building geometries varying in plan dimensions, eaves height, roof slope, wind direction, exposure condition and various degrees of leakage for internal pressure measurements.

All of these parameters are included in the output naming convention, such that any parameter can be isolated.

Typical file name: ADW100o100S048axxxx.HDF

pos 1-3 identifies the originating testing facility

for example: 'ADW' refers to the Alan Davenport Wind Engineering Group

pos 4-6 identifies the roof slope in 12ths to 3 digits

for example: 100 is 1 in 12

300 is 3 in 12

025 is 1/4 in 12

pos 7 identifies the exposure

for example: 'o' refers to open country exposure (zo = 0.03m)

's' refers to suburban exposure (zo=0.3m)

pos 8-10 identifies the model scale to 3 digits

for example: 100 is a 1 in 100 scale model pos 11 identifies the leakage case

for example: 'D' is a distributed leakage

'S' is a small opening

'L' is a large opening

'N' is no opening

'B' is basic test

the locations of the openings are included in the attached autocad figures

pos 12-14 is the building eaves height in feet to 3 digits

for example: 012 is 12 feet

018 is 18 feet

pos 15 differentiates between models when all other naming parameters are the same

pos 16-19 is the wind angle in degrees to be read as xxx.x

for example: 1800 is 180.0 degrees

the definition of wind angle is included in the attached autocad figures

the .HDF identifies the data format as a Hierarchical Data File which is a standard file format for the exchange of scientific data.

****

Matlab Input Files:

All building models are provided with nine accompanying input files. These files are used in the main Matlab program for graphical displays of the models. Five graphical displays are presented in the main program:

1. An exploded view of the pressure tap.

2. A 3D view of the pressure tap layout.

3. A general exploded view of a low-rise building detailing the numbering of the model faces.

4. Two photographs of the model while being tested in the wind tunnel.

The input files used to create the exploded view and 3D view are as follows:

'flbco.inp' - coordinates of model extremities of exploded view.

Layout of file-

Column 1	Column 2
x coordinate (full scale feet)	y coordinate

'flwfl.inp' - matrix used to create lines to connect the building extremities of exploded view.

Layout of file-

Column 1	Column 2
Point 1	Point 2

'fltpc.inp' - coordinates for pressure tap layout of exploded view.

Layout of file-

Column 1	Column 2	Column 3	Column 4
Tap Number	Face Number	x coordinate (full scale feet)	y coordinate

'tdbco.inp' - coordinates of model extremities for 3D view.

Layout of file-

Column 1	Column 2	Column 3
x coordinate	y coordinate	z coordinate
(full scale feet)		

'tdwfl.inp' - matrix used to create lines to connect the building extremities for 3D view.

Layout of file-

Column 1	Column 2
Point 1	Point 2

'tdtpc.inp' - coordinates for pressure tap layout of exploded view.

Layout of file-

Column 1	Column 2	Column 3	Column 4	Column 5
Tap Number	Face Number	x coordinate	y coordinate	z coordinate
		(full scale feet)		

'pics1.inp' - A general exploded view of a low-rise building detailing the numbering of the model faces.

'pics2.inp' - Close-up photograph of model in wind tunnel.

'pics3.inp' - Photograph of model in wind tunnel.

****

NOTE: Face 0 refers to internal pressure taps.

Only the taps listed in the *.inp file contain useful information.

All other taps in the related *.HDF files are not useable.

****

> A_Wind_Angle_Units [1 4] Type: char

Deg.

> Wind_Angle [1 1] Type: double

60.000000

> A_Reference_Wind_Speed_Units [1 3] Type: char

FPS

> Reference_Wind_Speed [1 1] Type: double

44.445000

> Exposure_Name [1 12] Type: char

Open_Country

> A_Wind_Speed_Profile_Units [1 7] Type: char

FS_Feet

> A_Wind_Speed_Profile_Column_IDs [1 16] Type: char

X Y Z V Iu Iv Iw

> Wind_Speed_Profile [34 5] Type: double

1.270000 0.465000 22.188000 0.000000 0.000000

1.270000 0.443000 23.151000 0.000000 0.000000

2.540000 0.512000 21.448000 0.000000 0.000000

4.880000 0.561000 19.136000 0.000000 0.000000

7.320000 0.609000 18.628000 0.000000 0.000000

9.750000 0.635000 18.175000 0.000000 0.000000

12.190000 0.647000 17.608000 0.000000 0.000000

15.850000 0.664000 16.381000 0.000000 0.000000

19.510000 0.687000 16.448000 0.000000 0.000000

23.160000 0.714000 15.902000 0.000000 0.000000

26.670000 0.717000 15.563000 0.000000 0.000000

30.480000 0.713000 15.379000 0.000000 0.000000

40.640000 0.761000 15.178000 0.000000 0.000000

50.800000 0.781000 15.085000 0.000000 0.000000

66.040000 0.813000 14.962000 0.000000 0.000000

81.280000 0.845000 14.454000 0.000000 0.000000

92.710000 0.877000 12.947000 0.000000 0.000000

92.710000 0.877000 12.473000 0.000000 0.000000

92.710000 0.869000 12.527000 0.000000 0.000000

93.980000 0.885000 12.627000 0.000000 0.000000

96.320000 0.885000 12.567000 0.000000 0.000000

98.760000 0.895000 12.222000 0.000000 0.000000

101.190000 0.902000 12.178000 0.000000 0.000000

103.630000 0.909000 12.031000 0.000000 0.000000

107.290000 0.907000 12.115000 0.000000 0.000000

110.950000 0.923000 11.750000 0.000000 0.000000

114.600000 0.939000 10.837000 0.000000 0.000000

118.110000 0.937000 10.864000 0.000000 0.000000

121.920000 0.958000 10.128000 0.000000 0.000000

132.080000 0.972000 9.893000 0.000000 0.000000

142.240000 0.984000 8.995000 0.000000 0.000000

157.480000 1.000000 7.860000 0.000000 0.000000

172.720000 0.991000 7.736000 0.000000 0.000000

184.150000 0.973000 8.613000 0.000000 0.000000

> Roof_Slope_in_12 [1 1] Type: double

0.020833

> A_Building_Height_Ft_Units [1 7] Type: char

FS_Feet

> Building_Height_Ft [1 1] Type: double

12.000000

> A_Building_Length_Ft_Units [1 7] Type: char

FS_Feet

> Building_Length_Ft [1 1] Type: double

125.000000

> A_Building_Width_Ft_Units [1 7] Type: char

FS_Feet

> Building_Width_Ft [1 1] Type: double

80.000000

> Building_Scale [1 1] Type: double

100.000000

> Building_Leakage [1 11] Type: char

Distributed

> Roof_Height_Velocity_Ratio [1 1] Type: double

0.560000

> A_Roof_Height_Velocity_Units [1 3] Type: char

FPS

> Roof_Height_Velocity [1 1] Type: double

25.200000

> A_Building_Corners_3D_Units [1 7] Type: char

FS_Feet

> A_Building_Corners_3D_Column_IDs [1 5] Type: char

X Y Z

> Building_Corners_3D [10 3] Type: double

0.000000 0.000000 0.000000

0.000000 125.000000 0.000000

80.000000 125.000000 0.000000

80.000000 0.000000 0.000000

0.000000 0.000000 12.000000

0.000000 125.000000 12.000000

40.000000 125.000000 12.833000

80.000000 125.000000 12.000000

80.000000 0.000000 12.000000

40.000000 0.000000 12.833000

> A_Wire_Frame_Lines_3D_Units [1 7] Type: char

FS_Feet

> A_Wire_Frame_Lines_3D_Column_IDs [1 64] Type: char

Line Segments from/to corners as specified in Building_Corners_3D

> Wire_Frame_Lines_3D [15 2] Type: int32

1 2

1 5

1 4

2 6

2 3

10 5

10 9

10 7

8 7

8 9

8 3

4 9

4 3

5 6

6 7

> A_Tap_Coordinates_3D_Units [1 7] Type: char

FS_Feet

> A_Tap_Coordinates_3D_Column_IDs [1 22] Type: char

Tap no. Face no. X Y Z

> Tap_Coordinates_3D [445 5] Type: double

101.000000 3.000000 34.360000 125.000000 10.000000

102.000000 3.000000 34.360000 125.000000 5.000000

111.000000 3.000000 28.710000 125.000000 5.000000

.

.

.

2216.000000 0.000000 39.000000 62.500000 0.000000

4215.000000 0.000000 40.000000 62.500000 0.000000

4216.000000 0.000000 41.000000 62.500000 0.000000

> A_Building_Corners_Flat_Units [1 7] Type: char

FS_Feet

> A_Building_Corners_Flat_Column_IDs [1 5] Type: char

X Y Z

> Building_Corners_Flat [28 2] Type: double

0.000000 -20.833000

0.000000 -8.833000

40.000000 -8.000000

80.000000 -8.833000

80.000000 -20.833000

-20.833000 0.000000

-20.833000 125.000000

-8.833000 125.000000

-8.000000 125.000000

-8.000000 0.000000

-8.833000 0.000000

0.000000 0.000000

0.000000 125.000000

40.000000 125.000000

80.000000 125.000000

80.000000 0.000000

40.000000 0.000000

88.000000 0.000000

88.000000 125.000000

88.833000 125.000000

100.833000 125.000000

100.833000 0.000000

88.833000 0.000000

0.000000 133.833000

0.000000 145.833000

80.000000 145.833000

80.000000 133.833000

40.000000 133.000000

> A_Flat_Wire_Frame_Lines_Units [1 7] Type: char

FS_Feet

> A_Flat_Wire_Frame_Lines_Column_IDs [1 66] Type: char

Line Segments from/to corners as specified in Building_Corners_Flat

> Flat_Wire_Frame_Lines [31 2] Type: int32

1 2

2 3

3 4

4 5

5 1

6 7

7 8

8 9

9 10

10 11

11 6

8 11

12 13

13 14

14 15

15 16

16 17

17 12

14 17

18 19

19 20

20 21

21 22

22 23

23 18

20 23

24 25

25 26

26 27

27 28

28 24

> A_Flat_Tap_Coordinates_Units [1 7] Type: char

FS_Feet

> A_Flat_Tap_Coordinates_Column_IDs [1 20] Type: char

Tap no. Face no. X Y

> Flat_Tap_Coordinates [442 4] Type: double

101.000000 3.000000 34.360000 163.830000

102.000000 3.000000 34.360000 168.830000

111.000000 3.000000 28.710000 168.830000

112.000000 3.000000 28.710000 163.830000

113.000000 3.000000 23.070000 163.830000

114.000000 3.000000 23.070000 168.830000

207.000000 3.000000 17.430000 168.830000

208.000000 3.000000 17.430000 163.830000

4009.000000 3.000000 68.210000 168.830000

4010.000000 3.000000 68.210000 163.830000

4011.000000 3.000000 62.570000 163.830000

4012.000000 3.000000 62.570000 168.830000

4105.000000 3.000000 56.930000 168.830000

4106.000000 3.000000 56.930000 163.830000

4107.000000 3.000000 51.290000 163.830000

4108.000000 3.000000 51.290000 168.830000

4201.000000 3.000000 45.640000 168.830000

4202.000000 3.000000 45.640000 163.830000

4203.000000 3.000000 40.000000 163.830000

4204.000000 3.000000 40.000000 168.830000

> Test_Year [1 1] Type: int32

2000

> Test_Month [1 1] Type: int32

0

> Test_Day [1 1] Type: int32

11

> Test_Hour [1 1] Type: int32

14

> Test_Minute [1 1] Type: int32

29

> Processed_Year [1 1] Type: int32

2003

> Processed_Month [1 1] Type: int32

5

> Processed_Day [1 1] Type: int32

29

> Processed_Hour [1 1] Type: int32

15

> Processed_Minute [1 1] Type: int32

12

> A_Data_Sample_Rate_Units [1 5] Type: char

Hertz

> Data_Sample_Rate [1 1] Type: double

500.000000

> No_Of_Samples_Per_Tap [1 1] Type: int32

49792

> No_Of_Pressure_Taps [1 1] Type: int32

672

> Image_1 [750 1120 3] Type: uint8

> Image_2 [1500 2240 3] Type: uint8

> Image_3 [1500 2240 3] Type: uint8

> No_Of_TS_Chan [1 1] Type: int32

672

> Ts_Multiplier [1 1] Type: double

0.001000

> Ts_Data_Type [1 9] Type: char

integer*2

> A_Tap_Max_Min_Mean_RMS_Column_IDs [1 40] Type: char

Tap no. Cp Max Cp Min Cp Mean Cp RMS

> Tap_Max_Min_Mean_RMS [672 5] Type: double

101.000000 0.456000 -0.148000 0.050000 0.049000

102.000000 0.573000 -0.098000 0.053000 0.044000

103.000000 0.259000 0.169000 0.215000 0.009000

104.000000 0.270000 0.201000 0.237000 0.008000

105.000000 0.322000 0.253000 0.284000 0.007000

.

.

.

4209.000000 0.077000 -0.034000 0.013000 0.012000

4210.000000 0.880000 0.440000 0.645000 0.072000

4211.000000 0.098000 -0.051000 0.026000 0.018000

4212.000000 0.803000 0.181000 0.458000 0.106000

4213.000000 0.111000 -0.081000 0.007000 0.019000

4214.000000 1.023000 0.259000 0.593000 0.138000

4215.000000 0.109000 -0.047000 0.025000 0.017000

4216.000000 0.111000 -0.054000 0.026000 0.017000

> Tap_Position_List [1 672] Type: int32

101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 501 502 503 504 505 506 507 508 509 510 511 512 513

.

.

.

3614 3615 3616 3701 3702 3703 3704 3705 3706 3707 3708 3709 3710 3711 3712 3713 3714 3715 3716 3801 3802 3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814 3815 3816 3901 3902 3903 3904 3905 3906 3907 3908 3909 3910 3911 3912 3913 3914 3915 3916 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016 4101 4102 4103 4104 4105 4106 4107 4108 4109 4110 4111 4112 4113 4114 4115 4116 4201 4202 4203 4204 4205 4206 4207 4208 4209 4210 4211 4212 4213 4214 4215 4216