RPC® File Formats

Introduction

This document contains information on RPC III file formats. You can use the file format information to:

- Read and write RPC III files.
- Import non-RPC data into RPC III files.

Format information that is common to all RPC III files is provided at the start of the document. Format information that is specific to each file type is provided in separate sections. At present, this information is provided for time history and histogram file formats.

For information specific to time history file formats, refer to "Time History Files," from the above menu.

For information specific to histogram file formats, refer to "Histogram Files," from the above menu.

RPC III File Organization

Most RPC III files have the same basic format.
RPC III files are sequential, fixed length, 512-byte record files.

The files contain a standard header that is followed by data. The data headers describe the data content of the file. The data portion of the file stores the actual data. There is no end-of-file character in an RPC III binary file.

An exception to this organization is a configuration file, which only contains a header. Configuration files are included under time history files.

Figure 6-1 shows how a file is organized:

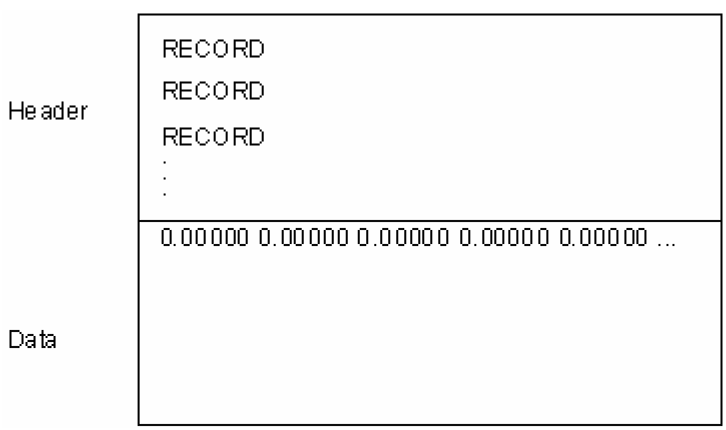


Figure 6-1. Organization of an RPC III File

RPC III File Headers

Header Function

RPC III file headers describe the data in the file. The headers also contain information needed to describe a specific file type.

NOTE Although an equal sign (=) is not entered when typing the keyword and associated value, it is included as part of a file header display.

Storage of Header Information

The ASCII file header contains information stored as records in 512-byte blocks. Each record in a block contains a keyword-value pair and is 128 bytes long. There are four records in each block.

NOTE If, as an example, a file header contained only ten records of keyword-value pairs, four of those records would be stored in block 1 of the header, four of them in block 2, and two in block 3. Block 3 would also include two records that did not include any keyword-value pairs. This header would only include 3 blocks of records.

The following example record contains the keyword FILE_TYPE and the value HISTOGRAM:

FILE_TYPE = HISTOGRAM

Bytes are allocated to a keyword-value pair as shown in Figure 6-2. There are 128 bytes in a record. A keyword is allocated a maximum of 32 bytes (including the null terminator). Each value is allocated a maximum of 96 bytes (including the null terminator).

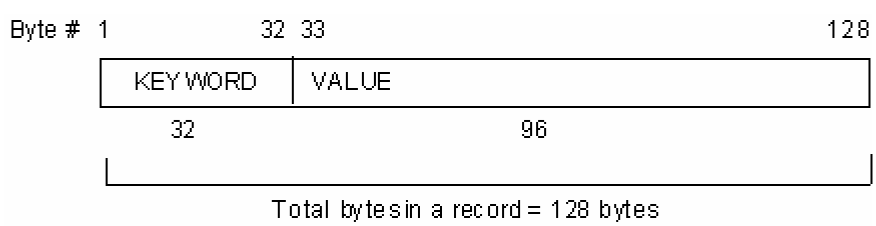


Figure 6-2. Allocation of a Keyword-Value Pair

The keyword-value pair character strings are stored in the header in a format that allows RPC III programs to access them. RPC III programs use byte arrays for string manipulation because the programs use both C and FORTRAN programming languages.

In a byte array, characters are stored as their ASCII values (in bytes), followed by a zero (0) that tells the program it has reached the end of the character string. For example, "C" is stored as 67, "h" is stored as 104, and so on. The array is dimensioned large enough to include the 0 used as a terminator.

For example, Figure 6-3 shows a null-terminated byte array that contains the word "Channel."

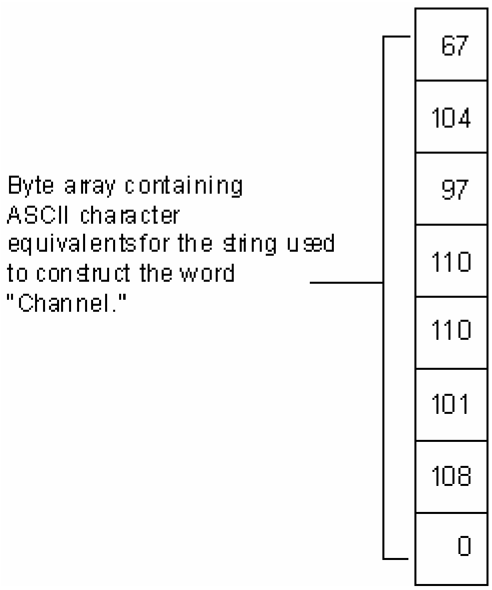


Figure 6-3. Null-Terminated Byte Array

Common record information

RPC III headers contain the following common records:

- Data storage format (binary or ASCII).
- Header size (in blocks).
- Number of parameters used by the file.
- File type.
- Date of file creation.
- Operation that created the file.

Table 6-1 describes the keywords and values associated with the common records.

Table 6-1. Keywords and Values Associated with Common Records

Keyword	Description	Possible Values	Position
FORMAT	Format in which data is stored.	Values are: BINARY_IEEE_LITTLE_END BINARY_IEEE_BIG_END BINARY ASCII ***NOTE*** The following binary values are processed the same way BINARY_IEEE_LITTLE_END BINARY	1
NUM_HEADER_BLOCKS	Number of 512-byte blocks used by the header information.	Up to 256 blocks	2
NUM_PARAMS	Total number of parameters (keyword-value pairs) in the file header.	Maximum number of parameters is 1024 (256 ´ 4).	3
FILE_TYPE	Type of data file.	File types are: Time history Configuration (subset of time history) Matrix Fatigue	Any

		Road surface Spectral Start	
DATE	Date and time the file or file header was created.	Windows NT date and time. Example: 6-May-96 15:14:45	Any
OPERATION	Name of program or operation that created the file.	Any RPC III software program or operation. Example: SINESW	Any

Record position

The first three records of any RPC III file header are position-sensitive. The FORMAT, NUM_HEADER_BLOCKS, and NUM_PARAMS keywords should always be in positions 1, 2, and 3 respectively in the header. All subsequent keywords can be placed in any position. The arbitrary placement of all but the first three keywords removes the need to count the number of bytes in order to determine a particular value. This structure also allows you to add a record to a file header without changing the way other files function.

However, because the records are position independent, certain keyword-value pairs may take longer to locate. For example, channel descriptors and unit descriptors may take a long time to locate. To reduce searching time, these should be listed in sequence in the file header.

Record maximum

There is no actual maximum for the number of records that can be placed in the header. Most RPC III programs create a temporary file to hold the header records before combining the header with the data. This temporary file (HEADER.TEMP) initially contains 1024 records. If the number of records in the header exceeds 1024, both the temporary file and the final data file are automatically extended. The number of records is limited only by the disk space available to hold both the header records and the data in one file. Similarly, if there are fewer than 1024 records, the size of the header is reduced when the header and data are combined in one file.

RPC III Data

RPC III data is placed directly after the header. Refer to individual sections for detailed information on how data is stored for each file type.

RPC III File Types

The RPC III software uses the following file types:

- TIME_HISTORY
- CONFIGURATION (Subset of TIME_HISTORY)
- MATRIX
- FATIGUE
- ROAD_SURFACE
- SPECTRAL
- START

A file type may contain different kinds of files. For example, there are seven kinds of time histories included under the TIME_HISTORY file type.

Each file included under a file type is identified in the header. For example, each type of time history is identified by the value in the TIME_TYPE keyword-value pair. A drive time history can have the value 1 or the value DRIVE, a response configuration file can have the value 6 or the value CONFIG_RESP. (Earlier versions of RPC III software use numbers.)

The following sections describe RPC III file formats and information specific to time history and histogram files.

Time History Files – Keywords

The header contains the keywords listed in Table 6-2:

Table 6-2. Keywords Included in Header

Keyword	**Description**	**Possible Values**
BYPASS_FILTER	Turns A/D filtering on or off.	Options: 0 Uses the filter (BYPASS_FILTER is off) 1 Does not use the filter (BYPASS_FILTER is on)
CHANNELS	Number of channels in the file.	Range of channels is 1 to 128.
DATA_TYPE	Numbers contained in the data of the file are of this type.	Data types are: SHORT_INTEGER (Default used by all RPC III programs) FLOATING_POINT (Not supported by RPC III V5.0x programs)
DATE	Date and time the file or file header was created.	NTFS date and time. Example: 22-Feb-2000 10:20:11 Format: dd-mm-yyy hh:mm:ss
DELTA_T	Time interval between consecutive points of data (in seconds).	A real number. Example: 4.882812E-03 Format is E8.6
DESC.CHAN_*n*	ASCII description of the specified channel. This description is repeated for each channel in the file.	If a user enters a description that contains more than one word separated by blanks, the user must enclose the words in quotation marks (" "). Maximum number of characters allowed is 96 if you output an RPC III file, 20 if you output an RPC II file. Example: "Left longitudinal axis"

FILE_TYPE	Type of data file.	File type is: TIME_HISTORY
FORMAT	Format in which data is stored.	Data storage formats are: BINARY_IEEE_LITTLE_END BINARY_IEEE_BIG_END BINARY ASCII
FRAMES	Number of frames of data stored in a file. A frame is set of data points ranging from 2 to any integer that is a power of 2. The maximum number of points in a frame is 8192.	Any valid number of frames.
HALF_FRAMES	Specifies whether a half frame is added to the beginning and end of a file to allow some forms of mathematical processing. The half frames contain no data although they increase the size of the file by one frame.	Options: 0 No half frames added 1 Half frames added
INT_FULL_SCALE	The maximum 16 bit integer value of the data	Default value is 32752 (which is 2^16 - 16)
LOWER_LIMIT.CHAN_*n*	Lower limit value defined for channel *n*.	Any valid number.
MAP.CHAN_*n*	Physical channel to which logical channel is mapped.	Any valid physical channel number. If you are working with multi-rate responses or a multi-test setup, channel mapping changes. For further information on the channel mapping changes, refer to RSURF program documentation in the *RPC III Program Reference Manual*.
NUM_HEADER_BLOCKS	Number of 512-byte blocks used by the header information.	Up to 256 blocks Example: 9
NUM_PARAMS	Total number of parameters (keyword-value pairs) in the file header.	Maximum number of parameters is 1024 (256 ´ 4).

OPERATION	Name of program or operation that created the file.	Any RPC III software program or operation. Example: SINESW
PARENT_*k*	File(s) from which this file was created. (*k* represents number of file.)	Any valid NTFS file name(s). Example: C:\TEST\DATA\POT.RSP
PART.CHAN_*n*	First channel assigned to partition *n*.	Any valid integer. Default value is 1.
PART.NCHAN_*n*	Number of consecutive channels assigned to partition *n*.	Any valid integer. Default value is the value in the keyword CHANNELS.
PARTITIONS	The number of groups of channels wanted. A partition is a submatrix within a matrix file. You should specify partitions if you anticipate that numerical operations on a time history data file will create a matrix file.	Range is 1 to 128 partitions. Default value is 1.
PTS_PER_FRAME	Integer that indicates number of points per frame of stored data. Must be a power of 2.	Options are 256, 512, 1024, and 2048 points and is a function of PTS_PER_GROUP. PTS_PER_FRAME must be £ PTS_PER_GROUP.
PTS_PER_GROUP	Total number of data points in the group. A group is a set of 2^n data frames in a channel. For example, if there are 4 frames in a group, then PTS_PER_GROUP is the sum of the points of those four frames: Frame 1 = 2048 points Frame 2 = 2048 points Frame 3 = 2048 points Frame 4 = 2048 points Points per group = 2048 ´4 = 8192 points	Integer sum of all points.
REPEATS	Number of times the frame is identically repeated within the file the first time the file is played out. This record is used for time history files that are used in frequency response function measurement.	Typically 1 repeat.

SCALE.CHAN_*n*	Value used to multiply the binary 16-bit converter value to find the engineering unit full-scale value. The value is calculated by dividing the engineering unit full scale by the integer full scale.	Example value: 3.053249E-04 Format is E8.6 (10.0 volts/32752.0 = 3.053249E-04)
TIME_TYPE	Type of time history. Older files (V4.0 and older) used the numeric representation. Newer files use the ASCII representation.	Time history types are: 1 DRIVE (drive file) 2 RESPONSE (response file) 3 MULT_DRIVE (multiplexed drive file) 4 MULT_RESP (multiplexed response file) 5 CONFIG_DRIVE (drive configuration file) 6 CONFIG_RESP (response configuration file) 7 PEAK_PICK (peak-picked file)
UNITS.CHAN_*n*	ASCII description of the engineering units associated with the channel. This description is repeated for each channel in the file.	Any valid unit. If a user enters a description that contains more than one word separated by blanks, the user must enclose the words in quotation marks (" "). Maximum number of characters allowed is 96 if you output an RPC III file, 20 if you output an RPC II file. Example value: "Micro Strain"
UPPER_LIMIT.CHAN_*n*	Upper limit value defined for channel *n*.	Any valid number.

Time History Headers

Example file

4CH.DRV is an example of a file provided by TAD, which contains both data and a time history file header. A TEDIT display of the first four frames of 4CH.DRV is shown in Figure 6-4.

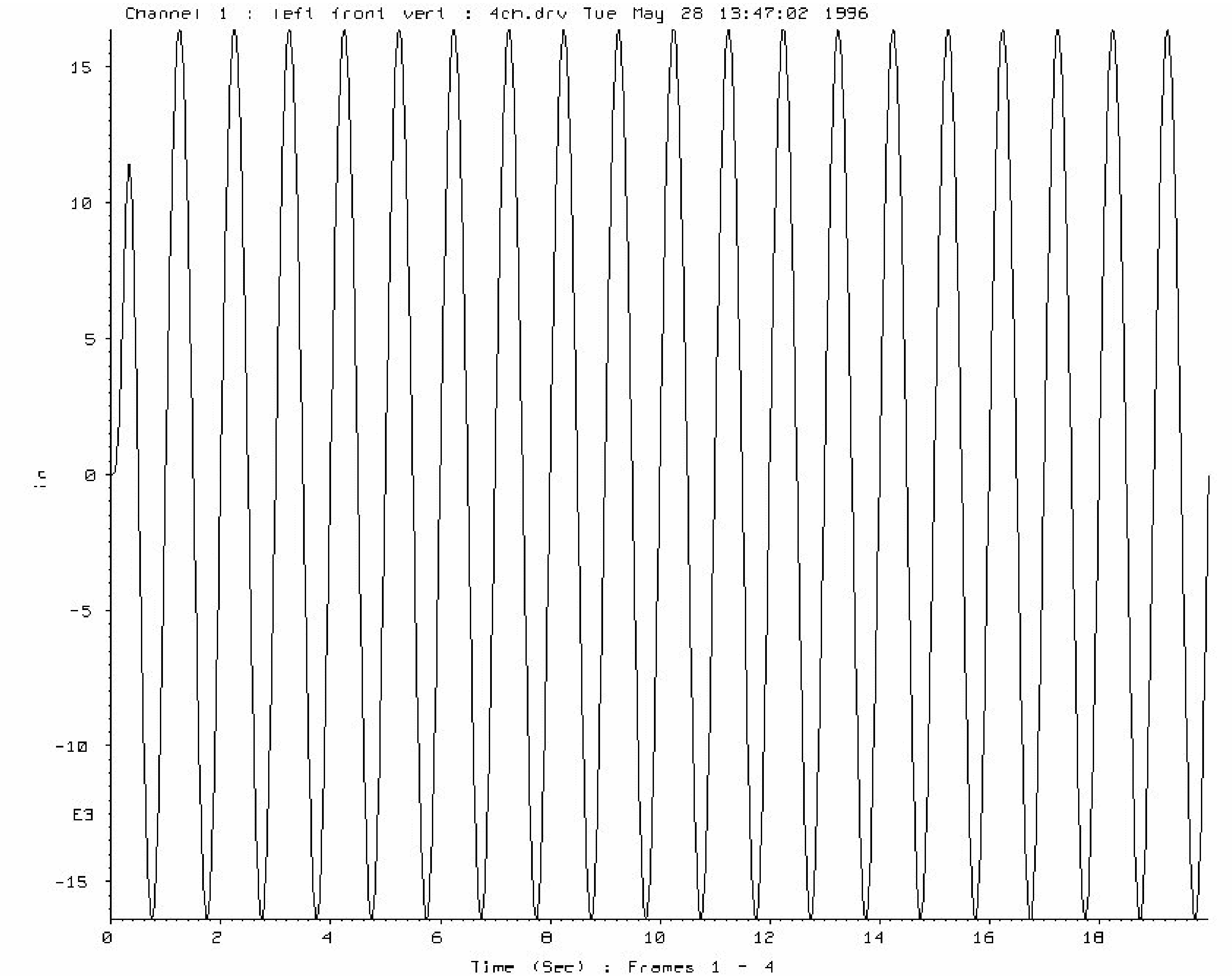


Figure 6-4. Start of 4CH.DRV

Example header

Either of two options are available for viewing the keyword-value pairs in a time history file header. You can use DUMP_RPC3 to display the header as shown in Figure 6-5.

Figure 6-5. Header from 4CH.DRV

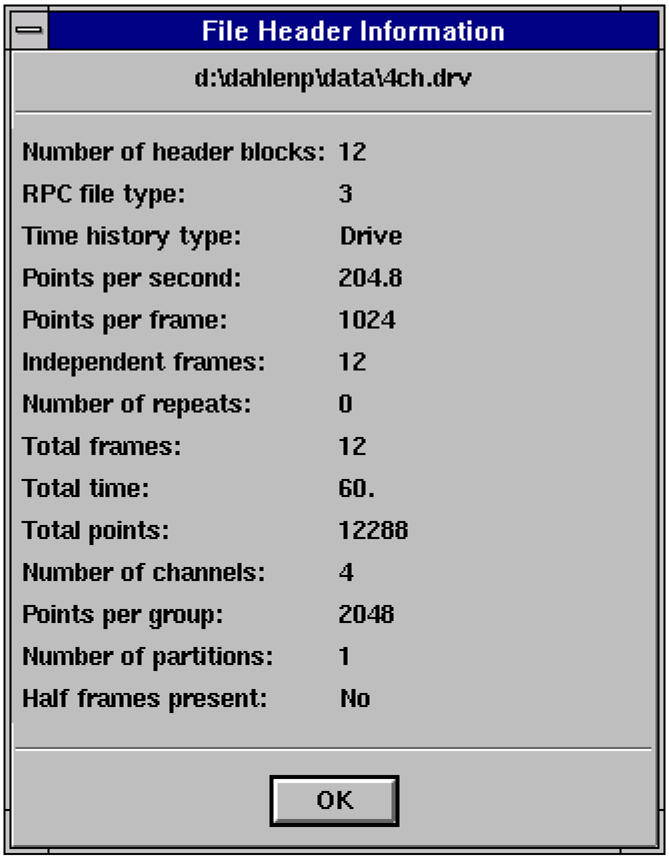
```
D:>dump_rpc3 4ch.drv

r DUMP.PARAMS YES

DUMP_RPC3 - MTS Remote Parameter Control Software - RELEASE 4.2A 24-MAY-1996

r LAST.USED d:\dahlenp\data\4ch.drv

w LAST.USED d:\dahlenp\data\4ch.drv

w LAST.USED d:\dahlenp\data\4ch.drv

r DEVICE.LISTING TERMINAL

r DEVICE.TERM xwin

w DEVICE.LISTING TERMINAL

FORMAT = BINARY_IEEE_LITTLE_END

NUM_HEADER_BLOCKS = 12

NUM_PARAMS = 45

FILE_TYPE = TIME_HISTORY

DATA_TYPE = SHORT_INTEGER

TIME_TYPE = DRIVE

DELTA_T = 4.882813E-03

PTS_PER_FRAME = 1024

CHANNELS = 4
```

```
PTS_PER_GROUP = 2048

BYPASS_FILTER = 0

HALF_FRAMES = 0

REPEATS = 0

FRAMES = 12

SCALE.CHAN_1 = 1.

UPPER_LIMIT.CHAN_1 = 1.

LOWER_LIMIT.CHAN_1 = -1.

MAP.CHAN_1 = 1

SCALE.CHAN_2 = 1.

UPPER_LIMIT.CHAN_2 = 1.

LOWER_LIMIT.CHAN_2 = -1.

MAP.CHAN_2 = 2

SCALE.CHAN_3 = 1.

UPPER_LIMIT.CHAN_3 = 1.

LOWER_LIMIT.CHAN_3 = -1.

MAP.CHAN_3 = 3

SCALE.CHAN_4 = 1.

UPPER_LIMIT.CHAN_4 = 1.

LOWER_LIMIT.CHAN_4 = -1.

MAP.CHAN_4 = 4

PARTITIONS = 1

PART.CHAN_1 = 1
```

```
PART.NCHAN_1 = 4

DESC.CHAN_1 = left front vert

UNITS.CHAN_1 = in

DESC.CHAN_2 = right front vert

UNITS.CHAN_2 = in

DESC.CHAN_3 = left rear vert

UNITS.CHAN_3 = in

DESC.CHAN_4 = right rear vert

UNITS.CHAN_4 = in

DATE = 05-Oct-95 14:19:14

OPERATION = TAD

PARENT_1 = d:\dahlenp\data\4ch_nt.drv

INT_FULL_SCALE = 32752
```

Figure 6-5. Header from 4CH.DRV

Or, you can select **File Header...** from the Show menu on the TEDIT window to display the following listing, which is not as complete as the one shown in Figure 6-5.



Time History Data

Groups and Frames RPC III places the data in time history files in groups, and it divides the groups of data into frames. The size of a frame can range from 256 to 16384 data points, while the size of a group can range from 2048 to 16384 data points. This grouping is referred to as demultiplexed blocks.

Demultiplexed data Figure 6-6 illustrates an example of the sequence used. There it is assumed that we are processing four-channels of time history data, which is divided into frames that contain 1024 data points, and groups that contain 4096 data points or four frames.

Note: Time History Data is stored in Single Precision floating point, which is 32 bit. The range is +/- 3.4028235E+38.

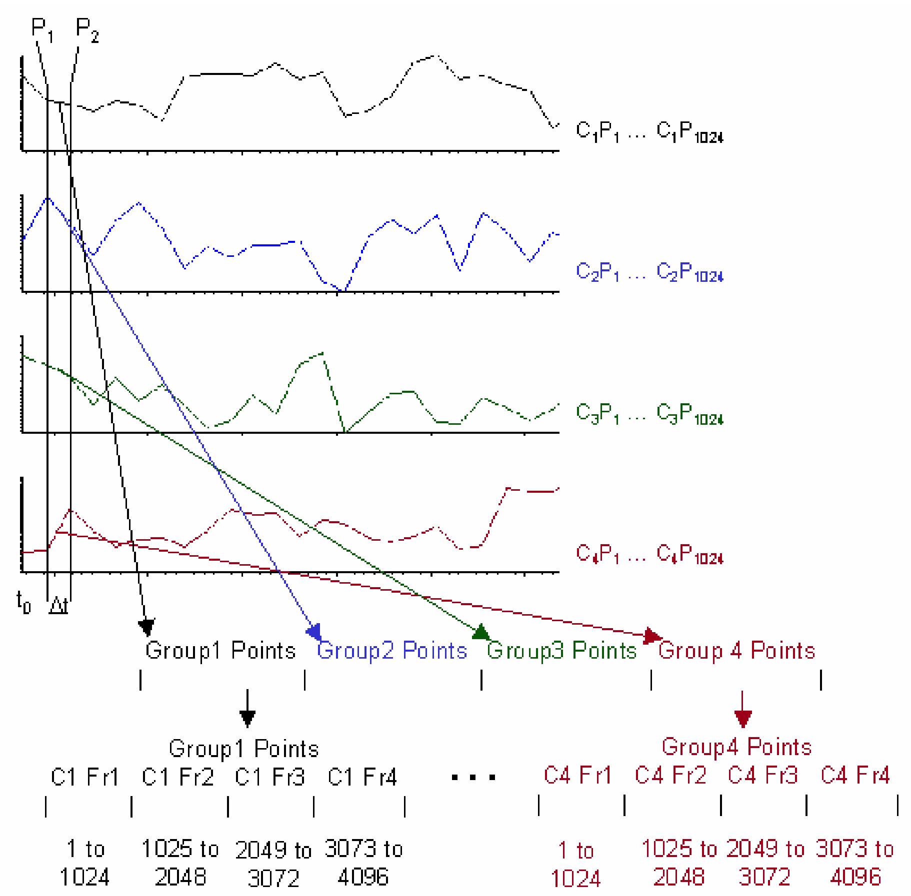


Figure 6-6. Data in a multiplexed block

Figure 6-6 represents the first block of a data file. The second block contains data points 4097 through 8194 of Channels 1 through 4. This sequence of blocks continues until the final block, which may not be completely filled with time history data.

Assume that there are 2304 data points remaining in Channels 1 through 4 for storage in the last block of the time history data file. That block is shown in Figure 6-7.

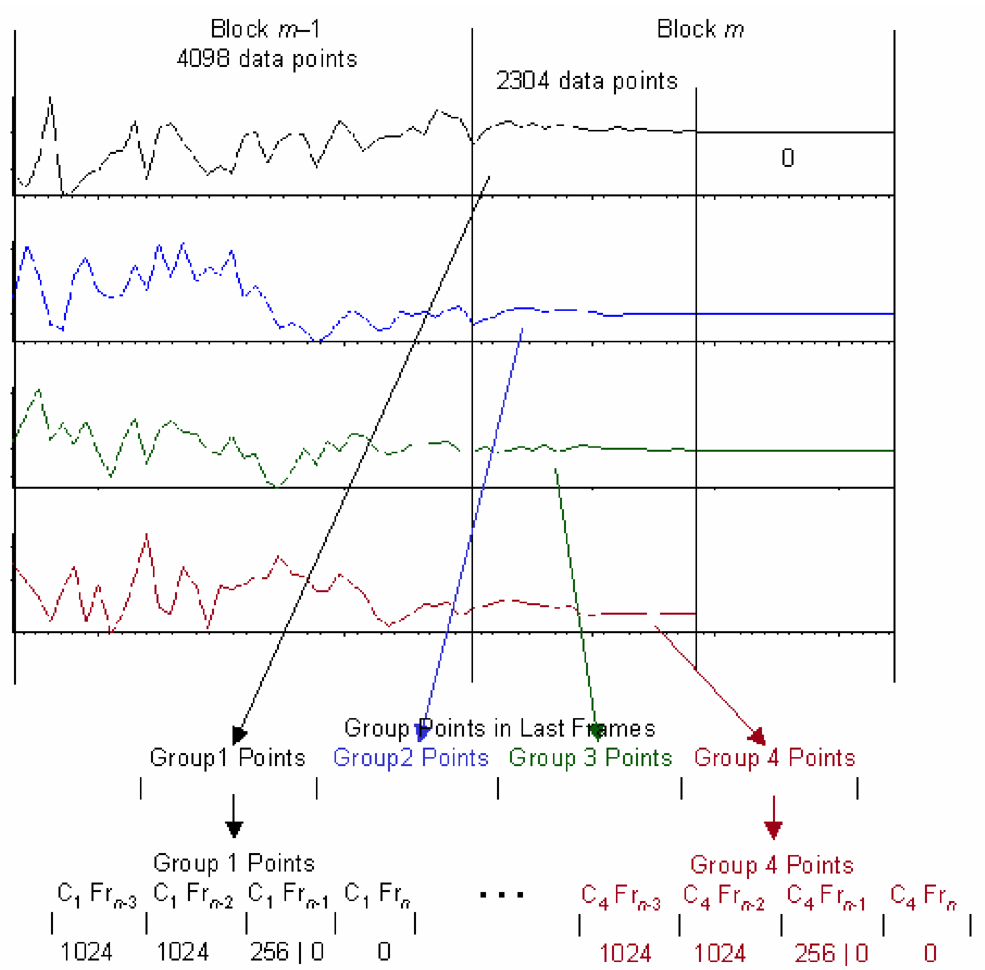


Figure 6-7. Data in last multiplexed block

In the last block of the example shown in Figure 6-7, the first and second frame for each channel includes 1024 points of time history data, while the third frame includes only 256 points of time history data. The remaining 768 data points in the third frame, and all 1024 data points in the fourth frame are set to 0.

Histogram Files – Keywords

Table 6-3 contains a list of all keywords used in histogram file headers.histogram file headers.

Table 6-3. Keywords Used in Histogram File Headers

Keyword	Description	Possible Values
ACTUALS_DEFINED	Whether the histogram actuals are defined.	YES Actuals are defined. NO Actuals are not defined.
CHANNELS	Number of channels in the file.	Range of channels is 1 to 128.
COL_ACTUALS.CHAN_*n*	The column variable actuals where *n* is the number of the channel. This data set contains the following members: Minimum Maximum	A data set of real values. Used when HISTOGRAM_TYPE is one of the following: • GENERIC_2D • GENERIC_3D • LEVEL_CROSS
COL_DESC.CHAN_*n*	ASCII description of the column variable for the specified channel *n*. This descriptor varies from one column to the next.	Any string. Maximum number of characters allowed is 96 (95 + null-byte terminator). A user enters this description when prompted to do so by a program. If a user enters a description that contains more than one word separated by blanks, the user must enclose the words in quotation marks (" "). Example: "Left longitudinal axis"

COL_PARMS.CHAN_*n*	The histogram column bounding set where *n* is the number of the channel. When HISTOGRAM_TYPE is GENERIC_2D or GENERIC_3D, this data set contains the following members: • Minimum • Maximum • Number of bins When HISTOGRAM_TYPE is LEVEL_CROSS, this data set contains the following members: • Minimum • Maximum • Reference • Levels	A data set containing values. The maximum, minimum, and reference number are reals. The number of bins and levels are integers. Used when HISTOGRAM_TYPE is one of the following: • GENERIC_2D • GENERIC_3D • LEVEL_CROSS
COL_UNITS.CHAN_*n*	ASCII description of the engineering units associated with the column variable for a specified channel *n*.	Any valid engineering unit value. Maximum number of characters allowed is 96 (95 + null-byte terminator). If a user enters a description that contains more than one word separated by blanks, the user must enclose the words in quotation marks (" "). Examples: Deg, mm, "micro strain"
DATE	Date and time the file was created.	Windows NT date and time. Example: 6-May-96 15:14:45
FILE_TYPE	Type of data file.	File type is HISTOGRAM.
FORMAT	Format in which data is stored.	Data storage format is: BINARY_IEEE_LITTLE_END or BINARY_IEEE_BIG_END
HISTOGRAM_DATA_TYPE	Bin data format (including overflow data).	Data type must be HIST_FLOAT.
HISTOGRAM_OFFSET.	Block pointer to start of data for the	Integer value.

CHAN_*n*	channel *n*. This value is referenced from the first header block which is 0 (not 1).	
HISTOGRAM_TYPE	The type of histogram.	Histogram types are histogram class-dependent. RANGE_MEAN RANGESTRAIN_ MEANSTRESS RANGESTRAIN_ MAXSTRESS RANGE MIN_MAX (minimums are rows, maximums are columns) If you select RAINFLOW_CYCLES or RAINFLOW_DAMAGE, valid histogram types are: If you select PD_COUNTS, valid histogram types are: GENERIC_3D GENERIC_2D If you select LX_COUNTS, the valid histogram type is LEVEL_CROSS
MAX_ACTUALS.CHAN_*n*	Maximum stress actuals for a specified channel *n*. A data set that contains the following members: Minimum Maximum	A data set of real values. Used when HISTOGRAM_TYPE is RANGESTRAIN_ MAXSTRESS
MAX_OVERFLOW. CHAN_*n*	Maximum stress overflow counts for a specified channel *n*. A data set that contains the following members: Under Over	A data set of real values. Used when HISTOGRAM_TYPE is RANGESTRAIN_ MAXSTRESS

MAX_PARMS.CHAN_*n*	The histogram maximum stress bounding set for a specified channel *n*. A data set that contains the following members: Minimum Maximum Number of bins	A data set containing values. The maximum and minimum are reals; the number of bins is an integer. Used when HISTOGRAM_TYPE is RANGESTRAIN_ MAXSTRESS
MEAN_ACTUALS.CHAN_*n*	Mean level actuals for a specified channel *n*. A data set that contains the following members: Minimum Maximum	A data set of real values. Used when HISTOGRAM_TYPE is one of the following: • RANGE_MEAN • RANGESTRAIN_ MEANSTRESS
MEAN_OVERFLOW. CHAN_*n*	Mean overflow counts for a specified channel *n*. A data set that contains the following members: Under Over	A data set of real values. Used when HISTOGRAM_TYPE is one of the following: • RANGE_MEAN • RANGESTRAIN_ MEANSTRESS
MEAN_PARMS.CHAN_*n*	The histogram mean bounding set for a specified channel *n*. A data set that contains the following members: Minimum Maximum Number of bins	A data set containing values. The maximum and minimum are reals; the number of bins is an integer. Used when HISTOGRAM_TYPE is one of the following: • RANGE_MEAN • RANGESTRAIN_ MEANSTRESS

MIN_MAX_ACTUALS. CHAN_*n*	The actual minimum and maximum values for a specified channel *n*. A data set that contains the following members: Minimum Maximum	A data set of real values. Used when HISTOGRAM_TYPE is MIN_MAX.
MIN_MAX_OVERFLOW. CHAN_*n*	The minimum and maximum overflow counts for a specified channel *n*. A data set that contains the following members: Under Over	A data set of real values. Used when HISTOGRAM_TYPE is MIN_MAX.
MIN_MAX_PARMS. CHAN_*n*	The histogram bounding set for a specified channel *n*. A data set that contains the following members: Minimum Maximum Number of bins	A data set that contains values. The maximum and minimum are reals; the number of bins is an integer. Used when HISTOGRAM_TYPE is MIN_MAX.
NUM_HEADER_BLOCKS	Number of 512-byte blocks used by the header information.	Up to 256 blocks. Example: 9
NUM_PARAMS	Total number of parameters (keyword-value pairs) in the file header.	Changes according to the number of channels in the file.
OPERATION	Name of program or process that created the file.	Any RPC III software program or operation. Example: DAMAGECYCLE
OVERFLOW_DEFINED	Whether histogram overflow and underflow bins are defined.	YES Overflow and underflow bins are defined NO Overflow and underflow bins are not defined
PARENT_*n*	File(s) from which this file was created.	Any valid NTFS file name(s).

RANGE_ACTUALS.CHAN_*n*	Range actuals for a specified channel *n*. This data set contains the following members: Minimum Maximum	A data set of real values. Used when HISTOGRAM_TYPE is one of the following: • RANGE_MEAN • RANGESTRAIN_ MEANSTRESS • RANGESTRAIN_ MAXSTRESS • RANGE
RANGE_OVERFLOW. CHAN_*n*	Range overflow counts for a specified channel *n*. This data set contains the following members: Under Over	A data set of real values. Used when HISTOGRAM_TYPE is one of the following: • RANGE_MEAN • RANGESTRAIN_ MEANSTRESS • RANGESTRAIN_ MAXSTRESS • RANGE
RANGE_PARMS.CHAN_*n*	The histogram range bounding set for a specified channel *n*. The range is the difference between successive peaks and valleys. This data set contains the following members: Minimum Maximum Number of bins	A data set of values. The maximum and minimum are reals; the number of bins is an integer. Used when HISTOGRAM_TYPE is one of the following: • RANGE_MEAN • RANGESTRAIN_ MEANSTRESS • RANGESTRAIN_ MAXSTRESS • RANGE
RESIDUALS.CHAN_*n*	Number of residuals (peaks and valleys from the unclosed cycles from a Rainflow cycle histogram).	Positive integer or zero (0). Zero means there are no unclosed cycles.

ROW_ACTUALS.CHAN_*n*	Row actuals for a specified channel *n*. This data set contains the following members: Minimum Maximum	A data set of real values. Used when HISTOGRAM_TYPE is GENERIC_3D.
ROW_DESC.CHAN_*n*	ASCII description of the row variable for a specified channel *n*. This descriptor varies from one row to the next.	A string. Maximum number of characters allowed is 96 (95 + null-byte terminator). If a user enters a description that contains more than one word separated by blanks, the user must enclose the words in quotation marks (" "). Example: "Left longitudinal axis"
ROW_PARMS.CHAN_*n*	The histogram bounding set for a specified channel *n*. This data set contains the following members: Minimum Maximum Number of bins.	A data set of values. The maximum and minimum are reals; the number of bins is an integer. Used when HISTOGRAM_TYPE is GENERIC_3D.
ROW_UNITS.CHAN_*n*	ASCII description of the engineering units associated with the row variable for a specified channel *n*.	Any valid engineering unit. Maximum number of characters allowed is 96 (95 + null-byte terminator). If a user enters a description that contains more than one word separated by blanks, the user must enclose the words in quotation marks (" "). Examples: Deg, mm, volts, "micro strain"

Histogram Headers

Example header Figure 6-8 shows an example of a histogram header:

Figure 6-8. Example of Histogram Header

```
d:\dahlenp>dump_rpc3 4ch.lvl

r DUMP.PARAMS YES

DUMP_RPC3 - MTS Remote Parameter Control Software - RELEASE 4.2A 24-MAY-1996

r LAST.USED d:\dahlenp\data\4ch.drv

w LAST.USED d:\dahlenp\data\4ch.lvl

w LAST.USED d:\dahlenp\data\4ch.lvl

r DEVICE.LISTING TERMINAL

r DEVICE.TERM xwin

w DEVICE.LISTING TERMINAL

FORMAT = BINARY_IEEE_LITTLE_END

NUM_HEADER_BLOCKS = 13

NUM_PARAMS = 49

FILE_TYPE = HISTOGRAM

HISTOGRAM_DATA_TYPE = FLOAT

HISTOGRAM_CLASS = LX_COUNTS

HISTOGRAM_TYPE = LEVEL_CROSS

ACTUALS_DEFINED = YES

OVERFLOW_DEFINED = NO
```

```
CHANNELS = 4

DATE = Mon Oct 09 13:00:54 1995

OPERATION = LEVELCROSS

PARENT_1 = d:\dahlenp\data\4ch.drv

ROW_DESC.CHAN_1 = N/A

COL_DESC.CHAN_1 = left front vert

ROW_UNITS.CHAN_1 = N/A

COL_UNITS.CHAN_1 = in

NOISEBAND.CHAN_1 = 0.

COL_PARMS.CHAN_1 = -1.091515e+004 1.091515e+004 0.000000e+000 3

COL_ACTUALS.CHAN_1 = -1.637600e+004 1.637600e+004

RESIDUALS.CHAN_1 = 0

DATA_OFFSET.CHAN_1 = 13

ROW_DESC.CHAN_2 = N/A

COL_DESC.CHAN_2 = right front vert

ROW_UNITS.CHAN_2 = N/A

COL_UNITS.CHAN_2 = in

NOISEBAND.CHAN_2 = 0.

COL_PARMS.CHAN_2 = -1.091515e+004 1.091515e+004 0.000000e+000 3

COL_ACTUALS.CHAN_2 = -1.637600e+004 1.637600e+004

RESIDUALS.CHAN_2 = 0

DATA_OFFSET.CHAN_2 = 14

ROW_DESC.CHAN_3 = N/A
```

```
COL_DESC.CHAN_3 = left rear vert

ROW_UNITS.CHAN_3 = N/A

COL_UNITS.CHAN_3 = in

NOISEBAND.CHAN_3 = 0.

COL_PARMS.CHAN_3 = -1.091515e+004 1.091515e+004 0.000000e+000 3

COL_ACTUALS.CHAN_3 = -1.637600e+004 1.637600e+004

RESIDUALS.CHAN_3 = 0

DATA_OFFSET.CHAN_3 = 15

ROW_DESC.CHAN_4 = N/A

COL_DESC.CHAN_4 = right rear vert

ROW_UNITS.CHAN_4 = N/A

COL_UNITS.CHAN_4 = in

NOISEBAND.CHAN_4 = 0.

COL_PARMS.CHAN_4 = -1.091515e+004 1.091515e+004 0.000000e+000 3

COL_ACTUALS.CHAN_4 = -1.637600e+004 1.637600e+004

RESIDUALS.CHAN_4 = 0

DATA_OFFSET.CHAN_4 = 16
```

Example data block The histogram data array and residual peak-valley sequence are stored back-to-back in contiguous 512-byte blocks. These blocks are ordered by rows and consist of a maximum of 128 reals (32-bit) per block.

The data blocks shown in Figure 6-9 are repeated for each channel.

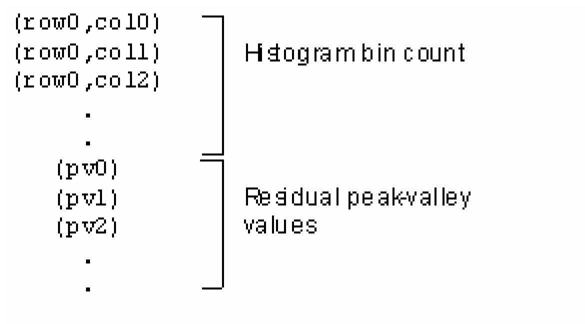


Figure 6-9. Data Blocks Storing Histogram Data Array and Residual

NOTE There will typically be some unused bytes in the last block written for each channel.

The block to start the write is referenced from the first block (which is 0).

Figure 6-10 shows an example of how the data is organized using a histogram with four bins. The histogram has a residual peak-valley sequence of −237.3, +992.6.

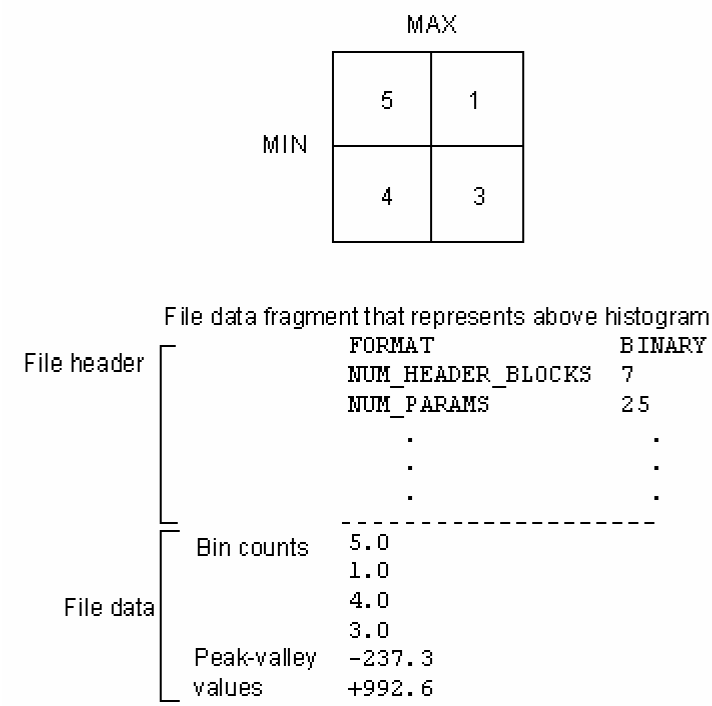


Figure 6-10. Example of How Data is Organized

Histogram Type Data Sets

Each histogram type and the keywords used by each type are listed next. They are shown in Table 6-4.

Table 6-4. Each Histogram Type and Associated Keywords

Histogram Type	Histogram Keywords
RANGE_MEAN or RANGESTRAIN_MEANSTRESS	RANGE_PARMS.CHAN_*n* MEAN_PARMS.CHAN_*n* RANGE_ACTUALS.CHAN_*n* MEAN_ACTUALS.CHAN_*n* RANGE_OVERFLOW.CHAN_*n* MEAN_OVERFLOW.CHAN_*n*
RANGESTRAIN_MAXSTRESS	RANGE_PARMS.CHAN_*n* MAX_PARMS.CHAN_*n* RANGE_ACTUALS.CHAN_*n* MAX_ACTUALS.CHAN_*n* RANGE_OVERFLOW.CHAN_*n* MAX_OVERFLOW.CHAN_*n*
MIN_MAX	MIN_MAX_PARMS.CHAN_*n* MIN_MAX_ACTUALS.CHAN_*n* MIN_MAX_OVERFLOW.CHAN_*n*

RANGE	RANGE_PARMS.CHAN_*n* RANGE_ACTUALS.CHAN_*n* RANGE_OVERFLOW.CHAN_*n*
GENERIC_2D	COL_PARMS.CHAN_*n* COL_ACTUALS.CHAN_*n*
GENERIC_3D	COL_PARMS.CHAN_*n* ROW_PARMS.CHAN_*n* COL_ACTUALS.CHAN_*n* ROW_ACTUALS.CHAN_*n*
LEVEL_CROSS	COL_PARMS.CHAN_*n* COL_ACTUALS.CHAN_*n*

Rules for Working with Histogram Files

The following rules control use of the HISTOGRAM_CLASS and HISTOGRAM_SUBCLASS_n header keyword-value pairs. The rules should help you avoid performing incorrect histogram operations such as adding damage histograms to cycles histograms. The rules also allow programs like HISTPRINT and HISTPLOT to have titles such as "Rainflow Cycles Histogram" or "Worst Case Morrow Rainflow Damage Histogram."

- The HISTOGRAM_CLASS header keyword is used to control program operations.
- The HISTOGRAM_SUBCLASS_*n* keywords are used (where appropriate) to help identify the histogram. These keywords do not control program operations.

Input/output rules Fatigue programs (and controller processing) use the histogram input/output rules listed in Table 6-5:

Table 6-5. Histogram Input/Output Rules

Program	Input Rule	Output Rule
Controller real-time process	Not applicable.	Only creates a RAINFLOW_CYCLES histogram.
DAMAGECYCLE	Not applicable.	Only creates a RAINFLOW_CYCLES histogram.
DAMAGEHIST	Only accepts a RAINFLOW_CYCLES histogram.	Only creates a RAINFLOW_DAMAGE histogram, that has two subclasses; one for damage model (for example, SWT) and the other for damage scenario case (for example, WORST).
HISTACCUM	Only accepts RAINFLOW_CYCLES histograms (with unclosed residuals).	Only creates an accumulated RAINFLOW_CYCLES histogram.
HISTCONVERT	Only accepts RAINFLOW_CYCLES, RAINFLOW_DAMAGE, or PD_COUNTS (with HISTOGRAM_TYPE = GENERIC_3D) histograms as inputs.	Only creates RAINFLOW_CYCLES or RAINFLOW_DAMAGE histograms. The RAINFLOW_DAMAGE subclasses are passed as long as they make sense. This is true for all situations except for the RangeMean2Range conversion. The histogram scenario case must be set to GENERIC if the damage model is SWT or MORROW. This is done to avoid comparing two interpretations that are not equivalent. For example, a worst case RangeMean interpretation is not equivalent to a worst case range interpretation if mean or maximum stress is involved.
HISTPRINT and HISTPLOT	Accept all classes and subclasses.	Not applicable.
LEVELCROSS	Not applicable.	Only creates an LX_COUNTS histogram.
PDHIST	Not applicable.	Only creates a PD_COUNTS histogram.
RAINFLOW	Not applicable.	Only creates a RAINFLOW_CYCLES histogram.

Using Actuals and Overflow Values

Actuals and overflows

Histogram file headers may contain actuals and overflow information. The actuals are the actual data values in a time history (for example a maximum and minimum). The overflow data is the data collected when some cycles (or other counts) exceed the histogram boundaries during data collection and sorting. These cycles are counted in the overflow bins. The actuals and overflow values are only created when you manually define the histogram boundaries for a data collection session.

Using actuals and overflows

Placing the actuals and overflow data in the header gives you access to the information needed to reset the histogram bounding keywords to accommodate the overflow values. A program or a process can then be repeated to capture the overflow data in the histogram bins without generating further overflows.

Collecting histogram data

You collect histogram data as follows:

- Online, during iterations or a durability test. You use BTREND or BDAMAGE (optional) to collect histogram data online.

- Offline, after your durability test is complete. You use RAINFLOW, DAMAGEHIST, LEVELCROSS, and HISTACCUM to create histograms offline.

Online histogram data collection

When you collect histogram data online during a test, you have to define the histogram boundaries manually. This is because you define the histogram boundaries before you know what the actual boundary definition values such the maximum and minimum will be. Therefore, you have to define the boundaries based on an estimate of what you think the values for the boundaries will be.

If you set the boundary definition values too low, the software will place all values that exceed the boundary definition values in the overflow bins. The following rules apply:

- If the number of data overflows does not exceed the allowed maximum number of overflows, the software increases the size of the histogram to accommodate the overflow values after the test is complete.

- If the number of overflow values exceeds the allowed maximum number of overflows, the allowed overflows are placed in the histogram and the additional overflow data is kept after the test is complete. However, because all the overflow histogram data is not added to the histogram, the histogram is missing data and may be unusable.

If you set the boundary values too high (for example, to the full-scale values), your data may end up placed in too few bins to be usable.

Offline histogram data collection

When you collect histogram data offline, you can define the histogram boundaries automatically or manually.

The offline histogram software processes response files that already exist. Therefore, if you allow the software to automatically define the boundaries, there will be no overflow bins.

If you manually define the histogram boundaries, any values that fall outside the manually-defined limits are placed in overflow bins.

Programs that use actuals and overflows

The actuals and overflow keywords are only used in some of the derived histogram files. For example, they are not used by DAMAGEHIST. Therefore, they do not exist in all histogram file headers. Whether or not actuals and overflows are used by a particular histogram file is defined by the header keyword-values ACTUALS_DEFINED and OVERFLOW_DEFINED.

The rules governing the use of these keywords is as follows.

- Both actuals and overflows are defined in histogram files generated by RAINFLOW, DAMAGECYCLE, HISTACCUM, HISTCONVERT, and the controller real-time process.

NOTE The overflows must be zero for all inputs to HISTACCUM.

- Both actuals and overflows are undefined in histogram files generated by DAMAGEHIST and HISTMATH.
- Actuals are defined and overflows are undefined in PDHIST and LEVELCROSS.