B

Standard Data Format (SDF)

This appendix describes the organization of files that use the Standard Data Format (SDF). SDF files contain both measurement data and the settings of important instrument parameters when the data was taken.

For more information on SDF data see chapter 2 and chapter 6.

Records—the Basic SDF Units

An SDF file consists of a number of smaller units called records: one "master," called the File Header, followed by a variable number of additional records. A record contains a group of related parameters and data. For example, a record called the Channel Header contains all channel-specific information for one of the channels used in a measurement.

There are eleven common records defined for SDF files:

File Header:	Provides an index to the file.
Measurement Header:	Contains settings of measurement parameters.
Data Header:	Tells you how reconstruct one block of measurement results (x- and y-axis values for every point of every trace).
Vector Header:	Tells you which channel (or pair of channels) provided data for a single trace.
Channel Header:	Contains channel-specific information for one channel used in the measurement.
Scan Structure:	Tells you how vectors are organized in the Y-axis Data record when the measurement includes multiple scans of data.
Scan Big:	Extended scan header which tells how vectors are organized in the Y-axis data record when the measurement may include more than 32767 scans of data.
Scan Variable:	Contains a number that identifies every scan (scan time, RPM, or scan number).
Comment Header:	Contains text information associated with the file.
X-axis Data:	Contains the x-axis data needed to reconstruct any trace.
Y-axis Data:	Contains the y-axis data needed to reconstruct any trace.

Another record—the Unique record—makes the SDF file format flexible. The eight common record types define parameters that are common to many instruments and systems. However, a particular instrument or system may need to save and recall the states of additional parameters. These states can reside in Unique records.

Record Order

Each measurement file begins with two ASCII characters followed by the File Header. Fields in the File Header tell you how many additional records follow and their order in the file. The order of records is always the same (although some files may not contain all record types).

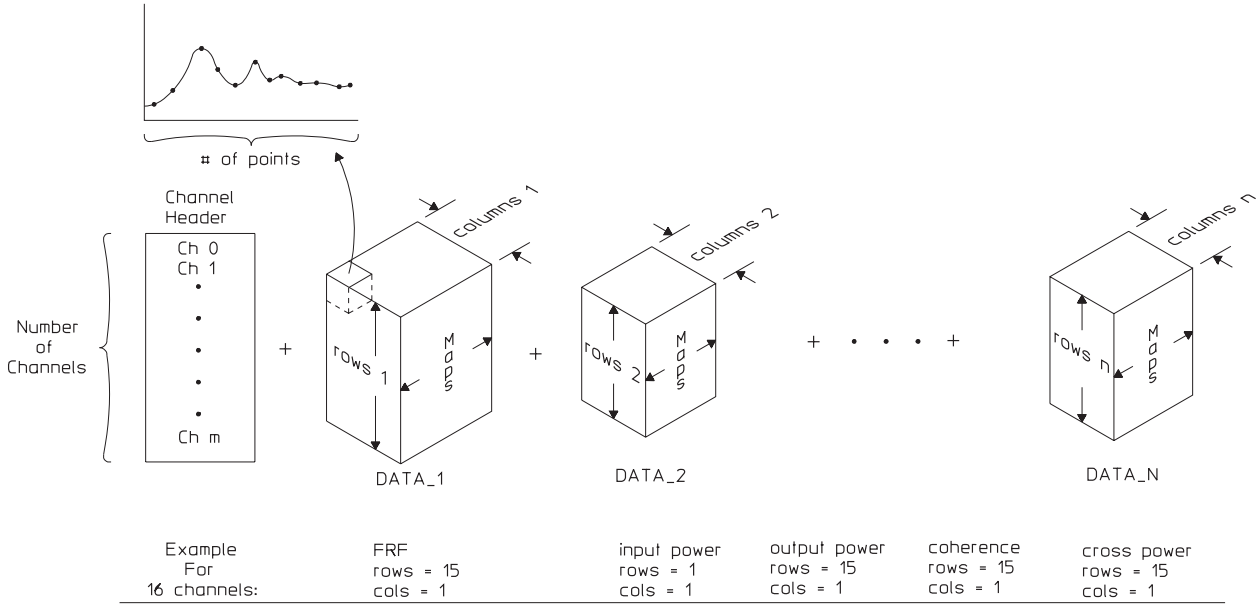

Figure B-5. Conceptual Model of an SDF File

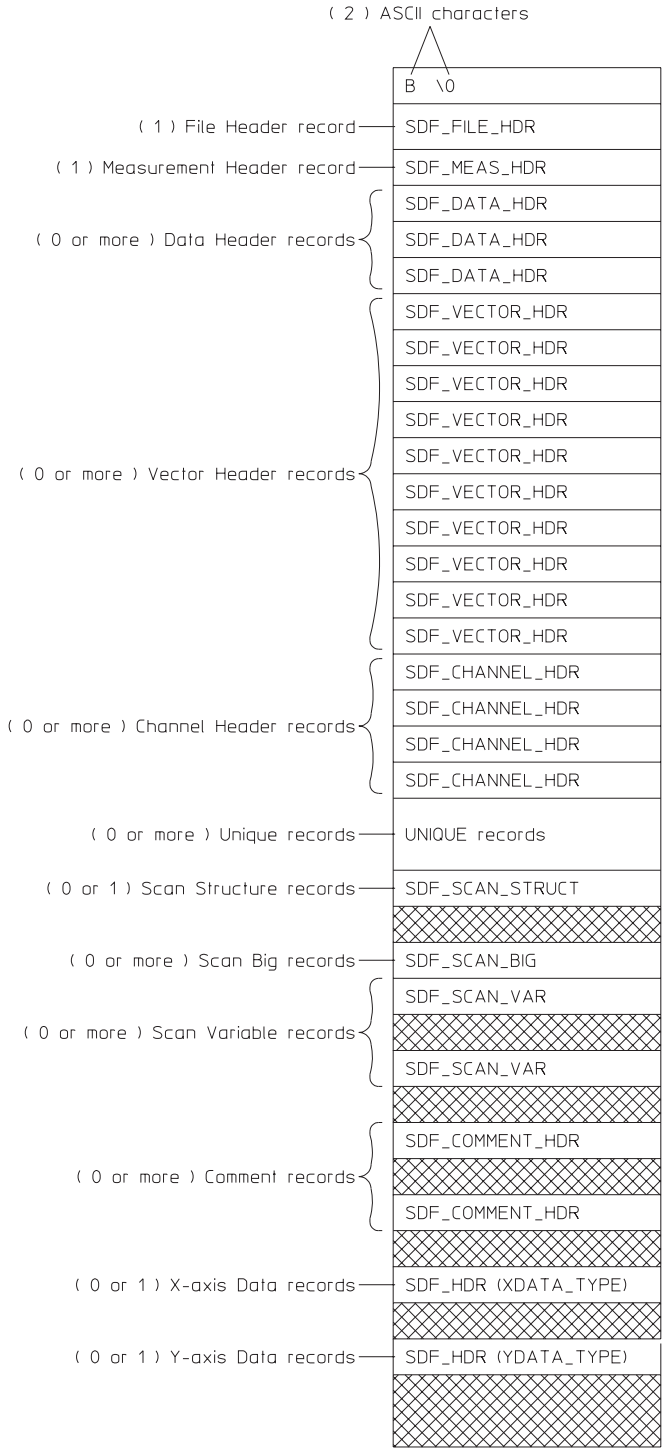

Figure B-6. SDF File Structure

The two ASCII characters at the beginning of the file identify it as an SDF file. The first character is B, the second is \0 (null).

Special Fields in a Record

All records contain two special fields:

recordType: A number that uniquely identifies the type of data contained in the record.

recordSize: A number that specifies the length of the record (in bytes).

The following table shows you the recordType and recordSize values for the eight common records:

Table B-1. Common Records

Name	Description	recordType	SDF 1 recordSize (bytes)	SDF 2 recordSize (bytes)	SDF 3 recordSize (bytes)
SDF_FILE_HDR	File Header	10	64	64	80
SDF_MEAS_HDR	Measurement Header	11	102	140	156
SDF_DATA_HDR	Data Header	12	114	134	148
SDF_VECTOR_HDR	Vector Header	13	18	18	18
SDF_CHANNEL_HDR	Channel Header	14	146	192	212
SDF_SCAN_STRUCT	Scan Structure	15	Variable	Variable	Variable
SDF_XDATA_HDR	X-axis Data	16	Variable	Variable	Variable
SDF_YDATA_HDR	Y-axis Data	17	Variable	Variable	Variable
SDF_SCAN_BIG	Scan Big	18	N/A	N/A	20
SDF_SCAN_VAR	Scan Variable	19	N/A	N/A	Variable
SDF_COMMENT_HDR	Comment Header	20	N/A	N/A	Variable

Record and Structure Descriptions

The following tables describes the individual records and structures from which the SDF files are built. Each table includes:

- A field index.
- A binary index.
- A description of each field.
- A description of the data type used in each field.
- The range of values for data in the field.

The fields in SDF records and structures use the following data types:

Char[n]:	This data type consists of a series of ASCII-encoded bytes. [n] specifies the number of bytes in the field.
Short:	This data type is used for integers (maximum range is –32768 to +32767). Values are transferred as two-byte, binary encoded integers.
Long:	This data type is also used for integers (but the maximum range is –2,147,483,648 to +2,147,483,647). Values are stored as four-byte, binary-encoded integers.
Float:	This data type is used for single-precision fixed-point and floating-point numbers. Values are stored in the 32-bit binary floating-point format.
Double:	This data type is used for double-precision fixed-point and floating-point numbers. Values are stored using the 64-bit binary floating-point format.

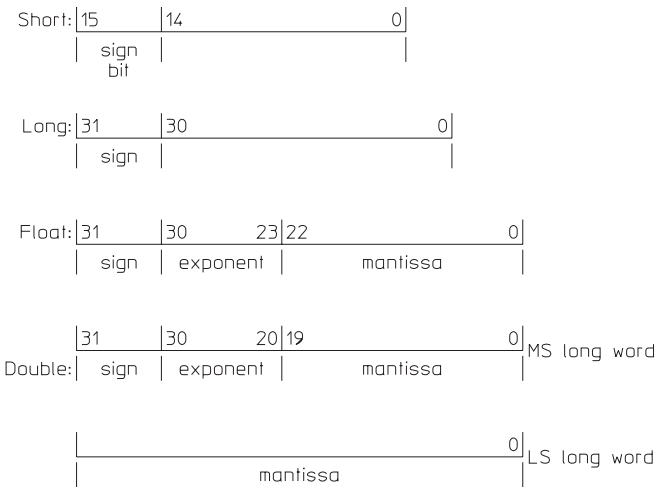

Figure B-7. IEEE Data Type Definition

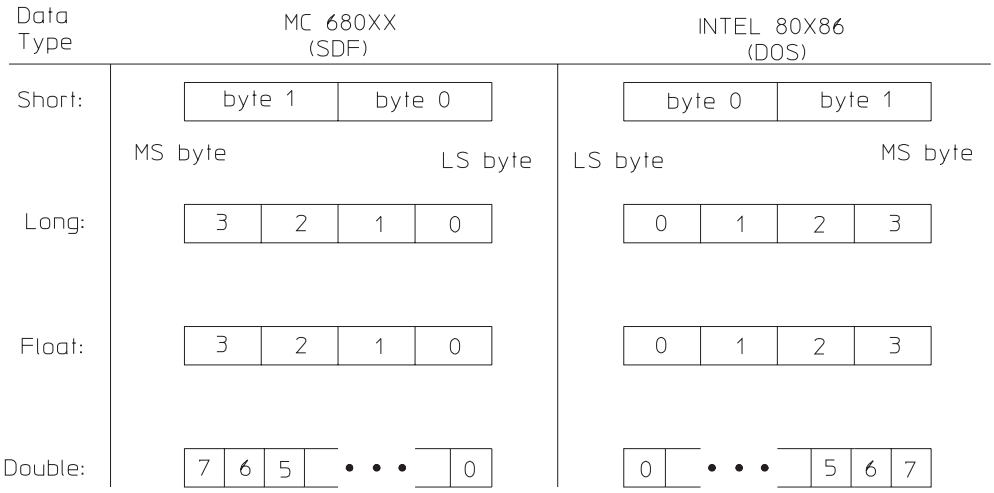

Figure B-8. SDF Field Data Types

Table B-2. SDF_FILE_HDR (File Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	10
2	3:6	recordSize	long	80 bytes
3	7:8	revisionNum — measurement file version number.	short	0:32767
4	9:10	applic — file saved from this instrument or application. –1=HP VISTA –2=HP SINE –3=HP 35660A –4=HP 3562A, HP 3563A – 5= HP 3588A – 6= HP 3589A – 99=unknown 1= HP 3566A, HP 3567A 2= HP 35665A 3= HP 3560A 4= HP 89410A, HP 89440A 7= HP 35635R 8= HP 35654A-S1A 9= HP 3569A 10= HP 35670A 11= HP 3587S	short	–99:32767
5	11:12	yearStamp — year at measurement start.	short	0:9999
6	13:14	monthDayStamp — month at measurement start. Encoded as (month ⟩100) + day For example, November 9 = 1109	short	0:1231
7	15:16	hourMinStamp — time at measurement start. Encoded as (hour ⟩100) + minute For example, 14:45 = 1445	short	0:2359
8	17:24	applicVer — software or firmware version number.	char[8]	i.d.
9	25:26	num_of_DATA_HDR_record — total Data Header records.	short	1:32767

* Bytes i.d. = instrument-dependent

Table B-2. SDF_FILE_HDR (File Header record), continued

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
10	27:28	num_of_VECTOR_record — total Vector Header records.	short	0:32767
11	29:30	num_of_CHANNEL_record — total Channel Header records.	short	0:32767
12	31:32	num_of_UNIQUE_record — total Unique records.	short	0:32767
13	33:34	num_of_SCAN_STRUCT_record — total Scan Structure records.	short	0:1
14	35:36	num_of_XDATA_record — total X-axis Data records.	short	0:1
15	37:40	offset_of_DATA_HDR_record — first Data Header record's byte offset from beginning of file.	long	$-1:(2^{31})-1$
16	41:44	offset_of_VECTOR_record — first Vector Header record's byte offset from beginning of file.	long	$-1:(2^{31})-1$
17	45:48	offset_of_CHANNEL_record — first Channel Header record's byte offset from beginning of file.	long	$-1:(2^{31})-1$
18	49:52	offset_of_UNIQUE_record — first Unique record's byte offset from beginning of file.	long	$-1:(2^{31})-1$
19	53:56	offset_of_SCAN_STRUCT_record — Scan Structure record's byte offset from beginning of file.	long	$-1:(2^{31})-1$
20	57:60	offset_of_XDATA_record — X-axis Data record's byte offset from beginning of file.	long	$-1:(2^{31})-1$
21	61:64	offset_of_YDATA_record — Y-axis Data record's byte offset from beginning of file.	long	$-1:(2^{31})-1$
22	65:66	num_of_SCAN_BIG_RECORD — total of SDF_SCAN_BIG and SDF_SCAN_VAR records.	short	0:32767
23	67:68	num_of_COMMENT_record — total of SDF_COMMENT_HDR records.	short	0:32767
24	69:72	offset_of_SCAN_BIG_record — the offset (from beginning of file) of the first Scan Big or Scan Variable record.	long	$-1:(2^{31})-1$
25	73:76	offset_of_next_SDF_FILE — allows more than one logical SDF FILE in a physical file. This supports multiple independent results taken at the same time. (For example, a time capture where the span and center frequencies of each channel are completely unrelated.) This offset points to the FORMAT_STRUCT record of the next logical SDF FILE in this physical file. All offsets in the next SDF FILE are relative to the start of the FORMAT field of the logical file (that is, ``B" followed by ``\0").	long	$-1:(2^{31})-1$

* Bytes i.d. = instrument-dependent

Table B-3a. SDF_MEAS_HDR (Measurement Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	11
2	3:6	recordSize	long	156 bytes
3	7:10	unique_record — byte offset from the beginning of the file to a record containing an instrument-specific measurement header. This field may be ignored when the file is recalled if it is recalled into an instrument type other than that used to create it.	long	$-1:0:(2^{31})-1$
4	11:14	centerFreqOld** — center frequency.	float	unit is Hz, range is i.d.
5	15:18	spanFreqOld** — frequency span.	float	unit is Hz range is i.d.
6	19:22	blockSize — number of time-domain samples taken. This field is only valid for FFT measurements.	long	
7	23:24	zoomModeOn — zoom mode (0=not zoomed, 1=zoomed). This field is only valid for FFT measurements.	short	0:1
8	25:26	startFreqIndexOld*** — the first alias-protected point on a frequency-domain trace.	short	0:last_valid_index (SDF_DATA_HDR)
9	27:28	stopFreqIndexOld*** — the last alias-protected point on a frequency-domain trace.	short	0:last_valid_index (SDF_DATA_HDR)
10	29:30	averageType 0=none 1=rms 2=rms exponential 3=vector 4=vector exponential 5=continuous peak hold 6=peak	short	0:6
11	31:34	averageNum — number of averages.	long	range is i.d.
12	35:38	pctOverlap — percentage of time-domain samples that are shared between successive time records. This field is only valid for FFT measurements.	float	number between 0 and 1
13	39:98	measTitle — measurement title or automath label.	char[60]	i.d.
14	99:102	videoBandWidth — tells you the bandwidth of the instrument's video filter. This field is only valid for swept spectrum measurements.	float	unit is Hz

* Bytes ** Prior to version 2.0 *** Prior to version 3.0 i.d. = instrument-dependent

Table B-3b. SDF_MEAS_HDR (Measurement Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
15	103:110	centerFreq — center frequency	double	unit is Hz range is i.d.
16	111:118	spanFreq — frequency span	double	unit is Hz range is i.d.
17	119:126	sweepFreq — current frequency for a swept measurement	double	unit is Hz range is i.d.
18	127:128	measType — measurement type –99 = unknown measurement 0 = spectrum measurement 1 = network measurement 2 = swept measurement 3 = FFT measurement 4 = orders measurement 5 = octave measurement 6 = capture measurement 7 = correlation measurement 8 = histogram measurement 9 = swept network measurement 10 = FFT network measurement	short	–99:10
19	129:130	realTime — whether the measurement was continuous in time 0 = not continuous 1 = continuous	short	0:1
20	131:132	detection — detection type –99 = unknown detection type 0 = sample detection 1 = positive peak detection 2 = negative peak detection 3 = rose-and-fell detection	short	–99:3
21	133:140	sweepTime — actual time for a swept measurement	double	unit is sec range is i.d.
22	141:144	startFreqIndex — the first alias-protected point on a frequency-domain trace. This field is only valid for FFT measurements, long	long	0:last_valid_index (SDF_DATA_HDR)
23	145:148	stopFreqIndex — the last alias-protected point on a frequency-domain trace. This field is only valid for FFT measurements, long	long	0:last_valid_index (SDF_DATA_HDR)
24	149:156	expAverageNum — number of exponential averages.	double	range is i.d.

* Bytes i.d. = instrument-dependent

Table B-4a. SDF_DATA_HDR (Data Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	12
2	3:6	recordSize	long	148 bytes
3	7:10	unique_record — byte offset from the beginning of the file to a record containing an instrument-specific data header. May be ignored if recalled into a different type instrument.	long	$-1:(2^{31})-1$
4	11:26	dataTitle — instrument- or user-supplied name for data type.	char[16]	i.d.
5	27:28	domain −99=unknown 0=frequency 1=time 2=amplitude 3=RPM 4=order 5=channel 6=octave	short	−99, 0:6
6	29:30	dataType −99=unknown 0=time 1=linear spectrum 2=auto-power spectrum 3=cross-power spectrum 4=frequency response 5=auto-correlation 6=cross-correlation 7=impulse response 8=ordinary coherence 9=partial coherence 10=multiple coherence 11=full octave 12=third octave 13=convolution 14=histogram 15=probability density function 16=cumulative density function, 17=power spectrum order tracking 18=composite power tracking 19=phase order tracking 20=rpm spectral 21=order ratio 22=orbit 23=HP 35650 series calibration 24=sine rms pwr data 25=sine variance data 26=sine range data 27=sine settle time data 28=sine integ time data 29=sine source data 30=sine overload data 31=sine linear data 32=synthesis 33=curve fit weighting function 34=frequency corrections (for capture) 35=all pass time data 36=norm reference data	short	−99, 0:76

* Bytes i.d. = instrument-dependent

Table B-4b. SDF_DATA_HDR (Data Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
6 (cont.)		dataType (continued) 37=tachometer data 38=limit line data 39=twelfth octave data 40=S11 data 41=S21 data 42=S12 data 43=S22 data 44=PSD data 45=decimated time data 46=overload data 47=compressed time data 48=external trigger data 49=pressure data 50=intensity data 51=PI index data 52=velocity data 53=PV index data 54=sound power data 55=field indicator data 56=partial power data 57=Ln 1 data 58=Ln 10 data 59=Ln 50 data 60=Ln 90 data 61=Ln 99 data 62=Ln user data 63=T20 data 64=T30 data 65=RT60 data 66=average count data 68= IQ measured time 69= IQ measured spectrum 70= IQ reference time 71= IQ reference spectrum 72= IQ error magnitude 73= IQ error phase 74= IQ error vector time 75= IQ error vector spectrum 76= symbol table data		
7	31:32	num_of_pointsOld***	short	
8	33:34	last_valid_indexOld***	short	
9	35:38	abscissa_firstXOld**	float	
10	39:42	abscissa_deltaXOld**	float	

* Bytes ** Prior to version 2.0 *** Prior to version 3.0 i.d. = instrument-dependent

Table B-4c. SDF_DATA_HDR (Data Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
11	43:44	xResolution_type — tells you how to find x-axis values for this Data Header record's traces. 0=linear — calculate values from abscissa_firstX and abscissa_deltaX 1=logarithmic — calculate values from abscissa_firstX and abscissa_deltaX 2=arbitrary, one per file — find values in the X-axis Data record; same vector used for every trace in the measurement file 3=arbitrary, one per data type — find values in the X-axis Data record; same x-axis vector used for each trace associated with this record 4=arbitrary, one per trace — find values in the X-axis Data record; unique x-axis vector for each trace associated with this record	short	0:4
12	45:46	xdata_type — tells you the size and format of each x-axis value. 1=short (two-byte, binary-encoded integer) 2=long (four-byte, binary-encoded integer) 3=float (four-byte, binary floating-point number) 4=double (eight-byte, binary floating-point number) This field is only valid if xResolution_type is 2, 3, or 4.	short	1:4
13	47:48	xPerPoint — number of x-axis values per each trace point. This field is only valid if xResolution_type is 2, 3, or 4.	short	0:32767
14	49:50	ydata_type — tells you the size and format of each y-axis value. **NOTE**: If yIsComplex=1, both the real and imaginary components of each y value require the number of bytes specified here. 1=short (two-byte, binary-encoded integer) 2=long (four-byte, binary-encoded integer) 3=float (four-byte, binary floating-point number) 4=double (eight-byte, binary floating-point number)	short	1:4
15	51:52	yPerPoint — number of y-axis values per each trace point. **NOTE**: A value containing both real and imaginary components is still considered a single value.	short	0:32767
16	53:54	yIsComplex 0=each y value has only a real component 1=each y value has both a real and an imaginary component	short	0:1
17	55:56	yIsNormalized 0=not normalized 1=normalized (all y values fall between 0.0 and 1.0 and are unitless, for example coherence or power spectrum).	short	0:1
18	57:58	yIsPowerData 0=not power data (for example, linear spectrum) 1=power data (for example, auto-power spectrum)	short	0:1

*Bytes i.d. = instrument-dependent

Table B-4d. SDF_DATA_HDR (Data Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
19	59:60	yIsValid — (0=not valid, 1=valid)	short	0:1
20	61:64	first_VECTOR_recordNum — first Vector Header record belonging to this Data Header record	long	0:(num_of_ VECTOR_ record) – 1 (SDF_FILE_ HDR)
21	65:66	total_rows — used to determine the number of traces associated with this Data header record; just multiply total_rows by total_columns	short	1:32767
22	67:68	total_cols — used to determine the number of traces associated with this Data header record; just multiply total_rows by total_columns	short	1:32767
23	69:90	xUnit — engineering unit used for x-axis values.	struct	see SDF_UNIT
24	91:92	yUnitValid — yUnit field in this record is valid 0=use Channel Header record's engUnit field for y-axis units. 1=use this record's yUnit field for y-axis unit.	short	0:1
25	93:114	yUnit — engineering unit used for y-axis values.	struct	see SDF_UNIT
26	115:122	abscissa_firstX — x-axis value of first point. This field is only valid if xResolution_type is 0 or 1.	double	$\pm 10^{34}$
27	123:130	abscissa_deltaX — spacing between x-axis points. xn = x(n-1) + abscissa_deltaX (xResolution_type 0) xn = x(n–1) abscissa_deltaX (xResolution_type 1) This field is only valid if xResolution_type is 0 or 1.	double	$\pm 10^{34}$
28	131:132	scanData — indicates whether the data is scanned 0 = This data header is associated with non-scanned data 1= This data header is associated with the scan structure	short	0:1
29	133:134	windowApplied — indicates whether the windows indicated have already been applied to the data 0 = windows have not been applied 1 = windows have been applied	short	0:1
30	135:138	num_of_points — number of discrete points in each trace associated with this record.	long	$0:(2^{31})–1$
31	139:142	last_valid_index — last point containing valid data.	long	0:(num_of_points)–1
32	143:144	overSampleFactor — Usually 1 > 1 = the data has been low-pass filtered but not decimated.	short	1:32767
33	145:146	multiPassMode — ``Multi-pass'' refers to a mode where data for multiple frequency spans is interleaved. 0 = not multi-pass data 1..4 = multi-pass, corresponding the HP 3565 gate array modes 5 = future multi-pass mode	short	0:5

34	147:148	multiPassDecimations — > 0 = the number of decimations included in the multi-pass data.	short	0:32767

*Bytes i.d. = instrument-dependent

Table B-5. SDF_VECTOR_HDR (Vector Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	13
2	3:6	recordSize	long	18 bytes
3	7:10	unique_record — byte offset from the beginning of the file to a record containing an instrument-specific vector header. May be ignored if recalled into a different type instrument.	long	$-1:(2^{31})-1$
4	11:14	the_CHANNEL_record[2] — tells you which channel or channels provided data for this trace. Each element of this array contains an index to a Channel Header record. An element refers to no channel if the index value is –1.	short (2x)	–1, 0:32767
5	15:18	pwrOfChan[2] — tells you what exponent was applied to corresponding channel data to create this trace. (For example, pwrOfChan[0] is the exponent applied to the_CHANNEL record[0]'s data.) record [0] for row channel Chan [0] " " record [1] for column channel Chan [1] " " **NOTE**: You must divide each pwrOfChan value by 48 to obtain the true value of the exponent.	short (2x)	0:32767

*Bytes i.d. = instrument-dependent

Table B-6a. SDF_CHANNEL_HDR (Channel Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	14
2	3:6	recordSize	long	212 bytes
3	7:10	unique_record — byte offset from the beginning of the file to a record containing an instrument-specific vector header. May be ignored if recalled into a different type instrument.	long	$-1:(2^{31})-1$
4	11:40	channelLabel — channel documentation	char[30]	i.d.
5	41:52	moduleId — location of channel in instrument.	char[12]	
6	53:64	serialNum — instrument (or module) serial number.	char[12]	
7	65:88	window	struct	see SDF_ WINDOW
8	89:90	weight 0=no weighting 1=A-weighting 2=B-weighting 3=C-weighting	short	0:3
8	91:94	delayOld***	float	
9	93:96	range	float	unit is dBV, range is i.d. & includes overhead for scaling
10	99:100	direction −9=−TZ −8=−TY −7=−TX −3=−Z −2=−Y −1=−X 0=no direction specified 1=X 2=Y 3=Z 4=R (radial) 5=T (tangential — theta angle) 6=P (tangential — phi angle) 7=TX 8=TY 9=TZ	short	0:9
11	101:102	pointNum — test point on device under test	short	0:32676
12	103:104	coupling (0=DC, 1=AC)	short	0:1
13	105:106	overloaded (0=no, 1=yes)	short	0:1
14	107:116	intLabel — label for the instrument's internal unit (such as V)	char[10]	i.d.
15	117:138	engUnit — engineering unit (EU) definition for this channel	struct	see SDF_UNIT
16	139:142	int2engrUnit — EU correction factor. Divide internal-unit data by this value to get EU data	float	$(10^{34}$ (except 0)

* Bytes *** Prior to version 3.0 i.d. = instrument-dependent

Table B-6b. SDF_CHANNEL_HDR (Channel Header record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
17	143:146	inputImpedance — Input impedance	float	unit ohm, range i.d.
18	147:148	channelAttribute –99 = unknown attribute 0 = no attribute 1 = tach attribute 2 = reference attribute 3 = tach and reference attribute 4 = clockwise attribute	short	–99:3
19	149:150	aliasProtected 0 = data was not alias protected 1 = alias protected	short	0:1
20	151:152	digital 0 = analog input channel 1 = digital input channel	short	0:1
21	153:160	channelScale — see channelOffset below	double	unit is volts, range is i.d.
22	161:168	channelOffset — when the data type is ``short'' or ``long'' the following formula will convert the data to volts: Volts = channelOffset + (channelScale ⎩Ydata)	double	unit is volts, range is i.d.
23	169:176	gateBegin — Gated sweep start time	double	unit is sec, range is i.d.
24	177:184	gateEnd — Gated sweep stop time	double	unit is sec, range is i.d.
25	185:192	userDelay — User specified input channel time delay or line length (not trigger delay)	double	unit is sec, range is i.d.
26	193:200	delay — amount of time between trigger event and start of data collection	double	unit is sec, range is i.d.
27	201:208	carrierFreq — carrier frequency for demodulated data	double	unit is Hz, range is i.d.
28	209:210	channelNumber — zero-based channel number	short	0:32767
29	211:212	channelModule — zero-based channel module	short	0:32767

*Bytes i.d. = instrument-dependent

Table B-7. SDF_SCAN_STRUCT (Scan Structure record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	15
2	3:6	recordSize	long	variable
3	7:8	num_of_scan — number of times the instrument collected a complete set of x- and y-axis vectors for all scan-based data types.	short	1:(2^{15})–1
4	9:10	last_scan_index — index of the last valid scan	short	0:(num_of_ scan)–1
5	11:12	scan_type — tells you how the vectors from different scans are organized in the Y-axis Data record. 0=depth — all scans for the first data type's vectors followed by all scans for the second data type's vectors, and so on. 1=scan — all data type's vectors for the first scan followed by all data type's vectors for the second scan, and so on.	short	0:1
6	13:14	scanVar_type — tells you the size and format of each scan variable value. 1=short (two-byte, binary-encoded integer) 2=long (four-byte, binary-encoded integer) 3=float (four-byte, binary floating-point number) 4=double (eight-byte, binary floating-point number)	short	1:4
7	15:36	scanUnit — engineering unit used for scan variables	struct	see SDF_UNIT
		Scan variable values start here.		

*Bytes i.d. = instrument-dependent

Table B-8. SDF_SCANS_BIG (Scan big record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	18
2	3:6	recordSize	long	20
3	7:10	unique_record — byte offset from the beginning of the file to a record containing an instrument-specific scan big header. May be ignored if recalled into a different type instrument.	long	$-1:(2^{31})-1$
4	11:14	num_of_scan — number of times the instrument collected a complete set of x- and y-axis vectors for all scan-based data types.	long	$-1:(2^{31})-1$
5	15:18	last_scan_index — index of the last valid scan.	long	0:(num_of_ scan)–1
6	19:20	scan_type — tells you how the vectors from different scans are organized in the Y-axis Data record. 0=depth — all scans for the first data type's vectors followed by all scans for the second data type's vectors, and so on. 1=scan — all data type's vectors for the first scan followed by all data type's vectors for the second scan, and so on.	short	0:1

Table B-9. SDF_SCAN_VAR (Scan variable record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	19
2	3:6	recordSize	long	variable
3	7:10	unique_record — byte offset from the beginning of the file to a record containing an instrument-specific scan variable header. May be ignored if recalled into a different type instrument.	long	−1:(2^{31})−1
4	11:14	headersize — size of the header portion of this record (excluding the scan variable values).	long	54
5	15:16	scanBase_type — type of scan variable 0 = unknown 1 = scan number 2 = time 3 = RPM 4 = temperature 5 = tachometer count	short	0:5
6	17:18	scanOrder_type — progression of scan values 0 = unknown 1 = increasing in value 2 = decreasing in value	short	0:2
7	19:20	DATA_recordNum— SDF_DATA_HDR record number associated with this record, (−1 if no specific association).	short	0:num_of_ DATA_HDR_ record −1
8	21:30	scan_ID — name of scan information.	char[10]	i.d.
9	31:32	scanVar_type — tells you the size and format of each scan variable value 1 = short (two-byte binary-encoded integer) 2 = long (four-byte binary-encoded integer) 3 = float (four-byte floating-point number) 4 = double (eight-byte floating-point number)	short	1:4
10	33:54	scanUnit — engineering unit used for scan.	struct	see SDF_UNIT
		Scan variable values start here.		

Table B-10. SDF_COMMENT_HDR (Comment record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	20
2	3:6	recordSize	long	variable
3	7:10	unique_record — byte offset from the beginning of the file to a record containing an instrument-specific comment header. May be ignored if recalled into a different type instrument.	long	$-1:(2^{31})-1$
4	11:14	headersize — size of the header portion of this record (excluding the comment text).	long	24
5	15:18	comment_bytes — size of comment (in bytes). This size may be smaller than the comment text area. If the size of the text is –1, then the entire comment text area is valid (or until an end-of-text marker is found).	long	–1: recordSize– headerSize
6	19:20	comment_type — type of comment data 0 = contains text	short	0:0
7	21:22	scope_type — tells which type of header the comment applies to 0 = entire file 1 = SDF_DATA_HDR 2 = SDF_VECTOR_HDR 3 = SDF_CHANNEL_HDR 4 = SDF_SCAN_STRUCT	short	0:4
8	23:24	scope_info — the index of the header associated with the scope_type (–1 = no specific header)	short	–1:32767
		Comment data starts here.		

Table B-11. SDF_XDATA_HDR (X-axis Data record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	16
2	3:6	recordSize	long	Variable
		X data values start here.		

*Bytes

Table B-12. SDF_YDATA_HDR (Y-axis Data record)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	recordType	short	17
2	3:6	recordSize	long	Variable
		Y data values start here.		

*Bytes

Table B-13. SDF_UNIT (Unit structure)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:10	label — engineering unit label (such as g).	char[10]	i.d.
2	11:14	factor — converts this unit to the equivalent SI unit	float	$(10^{34}$ (except 0)
3	15	mass — 2 times the unit exponent for the mass dimension	char	–128:127
4	16	length — 2 times the unit exponent for the length dimension	char	–128:127
5	17	time — 2 times the unit exponent for the time dimension	char	–128:127
6	18	current — 2 times the unit exponent for the current dimension	char	–128:127
7	19	temperature — 2 times the unit exponent for the temperature dimension	char	–128:127
8	20	luminal_intensity — 2 times the unit exponent for the luminal intensity dimension	char	–128:127
9	21	mole — 2 times the unit exponent for the mole dimension	char	–128:127
10	22	plane_angle — 2 times the unit exponent for the plane angle dimension	char	–128:127

*Bytes i.d. = instrument-dependent

Table B-14. SDF_WINDOW (Window structure)

Field Index	Binary Index*	Field Name/Description	Data Type	Range/ Units
1	1:2	windowType 0=window not applied 1=Hanning 2=Flat Top 3=Uniform 4=Force 5=Response 6=user-defined 7=Hamming 8=P301 9=P310 10=Kaiser-Bessel 11=Harris 12=Blackman 13=Resolution filter 14=Correlation Lead Lag 15=Correlation Lag 16=Gated 17=P400	short	0:12
2	3:4	windowCorrMode 0=correction not applied 1=narrow band correction applied 2=wide band correction applied	short	0:2
3	5:8	windowBandWidth **NOTE**: When windowType = 13, this field contains the instrument's resolution bandwidth.	float	unit is bins unit is Hz, range is i.d.
4	9:12	windowTimeConst — determines decay of Force and Response windows	float	unit is sec, range is i.d.
5	13:16	windowTrunc — width of FORCE window	float	unit is sec, range is i.d.
6	17:20	wideBandCorr — correction factor for wide-band signals (like random noise)	float	$\lceil 10^{34}$ (except 0)
7	21:24	narrowBandCorr — correction factor for narrow band signals (like sinesoidal wave)	float	$\lceil 10^{34}$ (except 0)

*Bytes i.d. = instrument-dependent

Reconstructing a Trace

The simplest SDF file defines a single trace. A trace consists of a series of discrete points, each defined by an X-axis value and a Y-axis value. This section tells you how to locate or calculate a single trace's X-axis and Y-axis values.

X-axis Values

A trace's X-axis values are stored in one of two SDF records: the Data Header record, or the X-axis Data record. The location used depends on whether or not consecutive points are evenly spaced along the X-axis:

Evenly-spaced points: X-axis values are stored in the Data Header record.
Unevenly-spaced points: X-axis values are stored in the X-axis Data record.

Field 11 of the Data Header record (xResolution_type) is an indicator of X-axis spacing. Consecutive points are evenly spaced if xResolution_type has value of 0 or 1; they are unevenly spaced if it has a value of 2, 3, or 4.

When consecutive trace points are evenly spaced along the X-axis (xResolution_type = 0, or 1), the SDF file does *not* include an X-axis Data record. Instead, the X-axis values of these points are completely defined by three fields in the Data Header record:

abscissa_firstX: tells you the X-axis value of the first point.

abscissa_deltaX: tells you the X-axis spacing between points.

num_of_points: tells you how many discrete points the trace is composed of.

You can calculate X-axis values this way when xResolution_type is 0 (linear resolution):

```
x0 = abscissa_firstX
x1 = x0 + abscissa_deltaX
x2 = x0 + abscissa_deltaX * 2
x3 = x0 + abscissa_deltaX * 3
   .
   .
   .
xn = x0  + abscissa_deltaX  * n  (where n = num_of_points  −1)
```

You can calculate X-axis values this way when xResolution_type is 1 (logarithmic resolution):

x0 = abscissa_firstX
x1 = x0 * abscissa_deltaX
$x2 = x0 * abscissa\_deltaX^2$
$x3 = x0 * abscissa\_deltaX^3$
.
.
.
$xn = x0 * abscissa\_deltaX^n$ (where n = num_of_points −1)

When consecutive trace points are *not* evenly spaced along the X-axis (xResolution_type = 2, 3, or 4), the SDF file includes an X-axis Data record. And when the file defines a single trace, that record includes just one vector of X-axis values.

Three fields in the Data Header record tell you how to read X-axis values from the X-axis Data record:

num_of_points: tells you how many discrete points the trace is composed of.

xdata_type: tells you the size and format of each X-axis value. (This information is encoded. For example, if xdata_type = 3, the size is four bytes and the format is binary floating-point.)

xPerPoint: tells you how many X-axis values are used to define a single point.

You can determine the size (in bytes) of the X-axis Data record's data block with the following formula:

num_of_points * xPerPoint * Size

The value of “Size” is encoded in xdata_type, as shown below:

xdata_type	Size (bytes)
1	2
2,3	4
4	8

Y-axis Values

A trace's Y-axis values are always stored in an SDF file's Y-axis Data record. And when the file defines a single trace, that record includes just one vector of Y-axis values. Reading this vector of values is similar to reading a vector of X-axis values from the X-axis Data record.

Four fields in the Data Header record tell you how to read Y-axis values from the Y-axis Data record:

num_of_points: tells you how many discrete points the trace is composed of.

ydata_type: tells you the size and format of each Y-axis value. (This information is encoded. For example, if ydata_type = 3, the size is our bytes and the format is binary floating-point.)

yPerPoint: tells you how many Y-axis values are used to define a single point.

yIsComplex: tells you whether each Y-axis value is purely real, or whether it consists of a real and an imaginary component.

You can determine the size (in bytes) of the Y-axis Data record's data block with the following formula:

$$\text{num\_of\_points} \times \text{yPerPoint} \times \text{Size} \times 2^{(\text{yIsComplex})}$$

The value of “Size” is encoded in ydata_type, as shown below:

ydata_type	Size (bytes)
1	2
2,3	4
4	8

Correcting Y-axis Data

Y-axis data is stored in the instrument's internal unit (usually volts) with all calibration corrections applied. Two other kinds of correction, however, are left for you to apply:

- Engineering unit (EU) correction.
- Window correction. (Necessary only for FREQ or ORDER domain data)

Note

Do not apply a window correction factor if it has already been applied by the instrument. A field in the Channel Header record — windowCorrMode — tells you whether or not the instrument has applied a window correction factor.

Also, do not apply a window correction factor if you are using the SdfYdata function; it will automatically apply these corrections to the data. See the chapter ``Using SDF Data with C Programs'' for more information on SdfYdata.

EU correction allows you to convert y-axis data from the instrument's internal unit to some user-defined unit (such as g — the acceleration of gravity). An EU correction factor is included in each Channel Header record; the factor's field name is int2engrUnit.

Window correction compensates for the shape of the resolution bandwidth filter in swept-tuned measurements. It compensates for the frequency-domain shape of the windowing function in FFT-based measurements. (FFT, which stands for Fast Fourier Transform, is an implementation of the Discrete Fourier Transform. Both are math algorithms that transform data from the time domain to the frequency domain.)

Two window correction factors are included in each Channel Header record's SDF_WINDOW structure:

wideBandCorr:	corrects for lost power when you are analyzing wide-band signals (like random noise).
narrowBandCorr:	corrects for lost amplitude when you are analyzing narrow-band signals (like a sinesoidal wave).

Check the windowCorrMode field (also in SDF_WINDOW) to find out if your instrument has already applied one of these factors to the Y-axis data. The field contains encoded values:

0:	no window correction.
1:	narrowBandCorr.
2:	wideBandCorr.

Role of the Vector Header

The Vector Header record contains the information you need to link a trace to the appropriate correction factors. First, it tells you which channels were used to gather raw data for the trace. It is the correction factor for these channels that you will apply to the trace. Second, it tells you how the raw channel data was processed to create the trace data. The correction factors must be processed in the same way before they are applied to the trace data. An example will help to illustrate this point.

A frequency response trace is created by processing the data from two channels in the following way:

$$\text{frequency response} = \frac{\text{response channel spectrum}}{\text{excitation channel spectrum}}$$

This formula can also be written as:

$$\text{freq resp} = (\text{resp ch spec})^{1} \times (\text{exci ch spec})^{-1}$$

A similar formula must be used to derive the correction factor for the frequency response trace:

$$\text{freq resp correction} = (\text{resp ch correction})^{1} \times (\text{exci ch correction})^{-1}$$

What values would a Vector Header record contain for this frequency response trace? Here is an example record:

recordType:	13
recordSize:	18
unique_record:	-1
the_CHANNEL_record[0]:	5
the_CHANNEL_record[1]:	2
pwrOfChan[0]:	48
pwrOfChan[1]:	-48

Two fields in the record — the_CHANNEL_record[0] and the_CHANNEL_record[1] — tell you which channels were used to measure response and excitation. In the example used here, these values indicate that the channels are described in Channel Header record 5 and Channel Header record 2. Two other fields — pwrOfChan[0] and pwrOfChan[1] — tell you what exponent was applied to each channel's data. (pwrOfChan[0] provides the exponent for the_CHANNEL_record[0] and pwrOfChan[1] provides the exponent for the_CHANNEL_record[1].)

Note 

You must divide each pwrOfChan value by 48 to obtain the true value of the exponent.

Numbers in the_CHANNEL_record fields are indexes to Channel Header records in the file (0 refers to the first one, 1 to the second, and so on). However, there is *no* correlation between a record's index number and the number of the channel described in that record. Channel Header record 0, for example, could describe Channel 2 of your instrument. To determine which channel is described in the record, you must read the record's channelLabel field.

Creating the Correction Factor

As you have seen in this section, the Vector Header record and its associated Channel Header records contain all the information you need to create a correction factor for a trace's Y-axis data. Just complete the following steps:

1. Determine which channels were used to gather data for the trace. (Look at the Vector Header record's the_CHANNEL_record fields.)
2. Determine which window correction factor (if any) was used for each channel. (Look in the Channel Header record's SDF_WINDOW structure; the field is windowCorrMode.)
3. Create a combined correction factor for each channel using the following formula:

$$\left\{\frac{\text{window correction}}{\text{EU correction}}\right\}^{\text{pwrOfChan}/48}$$

   window correction is equal to 1 if step 2 indicates that no window correction was used, otherwise it is the value of either narrowBandCorr or wideBandCorr.

   EU correction is the value of int2engrUnit.

4. Create a trace correction factor by multiplying the two channel correction factors.

Now you can correct the Y-axis data. Just multiply each Y-axis value by the trace correction factor.

Note

Some traces are created with data from just one channel. (To indicate that only one channel was used, one “the_CHANNEL_record” field contains a value of −1.) For traces such as these, the combined correction factor for the single channel is also your trace correction factor.

Locating a Vector of Y-axis Data

When an SDF file defines just one trace, the Y-axis Data record contains a single vector of Y-axis data. However, when the file defines more than one trace, the record contains many vectors — one for each trace. This section tells you how to locate one Y-axis vector in a block of many.

Single-Scan Files

When a file does *not* contain a Scan Structure record, the structure of the Y-axis data block is quite simple: the order of vectors in the block is exactly the same as the order of Vector Header records in the file. The vector described by Vector Header record 0 is first in the block, the one described by Vector Header record 1 is second, and so on.

Multiple-Scan Files

When a file does contain a Scan Structure record, the structure of the Y-axis data block is somewhat more complex:

- The block contains more than one set vectors.
- The vectors can be organized in two different ways.

A single scan puts a complete set of vectors in the Y-axis data block — one for each Vector Header record. Each additional scan adds another set of vectors to the block.

The two types of vector organization are referred to as Scan and Depth. The scan_type field in the Scan Structure record tells you which type of organization was used for your file. The following example shows how Scan and Depth differ.

The example file contains two Data Header records and seven Vector Header records (num_of_DATA_HDR_record and num_of_Vector_record in the File Header record have values of 2 and 7, respectively). Its Y-axis data block includes three scans of data (num_of_scan in the Scan Structure record has a value of 3). The first Data Header record is linked to the first three Vector Header records (total_rows ⎱total_cols = 3; first_VECTOR_recordNum = 0) and the second is linked to the last four (total_rows ⎱total_cols = 4; first_VECTOR_recordNum = 3).

Here are the two ways the vectors could be organized in the Y-axis data block:

scan_type is Scan (1):		**scan_type is Depth (0):**	
S0.D0.	V0	S0.D0.	V0
	V1		V1
	V2		V2
D1.	V3	S1.D0.	V0
	V4		V1
	V5		V2
	V6	S0.D1.	V3
S1.D0.	V0		V4
	V1		V5
	V2		V6
D1.	V3	S1.D1.	V3
	V4		V4
	V5		V5
	V6		V6

Time Capture Data

A time capture data contains long streams of raw time data captured by an instrument. The files contains some or all of the following results.

Time Data

Time data is identified in a time capture file with the following SDF_DATA_HDR attributes:

dataType	0 (time data)
ydata_type	SHORT16 or LONG32
scanData	1

The time data should always be the first result in a time capture file. To convert the data from its integer form to volts, you need to determine the channel header that belongs to the data and get the scaling factor from the channel header.

The time capture data is stored with multiple rows of data, one row for each channel of captured data. For a given row, the desired SDF_VECTOR_HDR number is (*first_VECTOR_recordNum* + row). After locating the SDF_VECTOR_HDR for this row, find the channel number from the_CHANNEL_record[0]. After locating the SDF_CHANNEL_HDR, you can scale the time capture data to volts by using the scale values in the SDF_CHANNEL_HDR. For each time capture value in the data result (defined as time_data[i]):

time_volts[i] = *channelOffset* + (*channelScale* ⎱time_data[i])

To determine the total number of time capture points, you need to know now many scans are in the file. The number of usable scans can be determined from the *last_scan_index* field in the SDF_SCAN_STRUCT. The number of usable points per scan can be determined from the last_valid_index in the SDF_DATA_HDR. The total number of time capture data points is:

time_capture_points = (*last_valid_index*+1) ⎱(*last_scan_index*+1)

Overload Data

Overload data is identified in a time capture file with the following SDF_DATA_HDR attributes:

dataType	46 (overload data)
ydata_type	SHORT16
num_of_points	1
scanData	1

This data result identifies whether an overload occurred in each scan of a time capture. There is one of these results for each channel of a time capture.

Frequency Correction Data

Frequency Correction Data is identified in a time capture file with the following SDF_DATA_HDR attributes:

dataType	34 (frequency corrections)
scanData	0

This data result contains the frequency domain corrections for the time capture data. It is the same size as the instrument's Spectrum data result. There is one of these results for each channel of a time capture.

Compressed Time Data

The compressed time data result in a time capture file contains a compressed representation of the entire time capture file of each channel of time data. The time data is sampled and minimum and maximum data values for each sampled region are stored for each data point. The compressed time data is identified in a time capture file with the following SDF_DATA_HDR attributes:

dataType	45 (decimated time data) or 47 (compressed time data)
ydata_type	SHORT16 or LONG32
yPerPoint	5
yIsComplex	0
scanData	0

The 5 values per point represent the following:

minimum real value

minimum imaginary value

maximum real value

maximum imaginary value

overload flag

Since the minimum/maximum values are in integer form, they can be scaled to volts using the same mechanism as for scaling the time data (see above). If the dataType is 45 (decimated time data), then the minimum values equal the maximum values since the data is decimated, not compressed. There is one of these results for each channel of a time capture.

Tachometer Data

Tachometer data is identified in a time capture file with the following SDF_DATA_HDR attributes:

dataType	37 (tachometer data)
ydata_type	LONG32
yIsComplex	0
scanData	0

Each point in the tachometer data is a 32 tachometer counter value which represents time. To interpret the tachometer data, you need to know how many tachometer points are in the tachometer data, the tachometer frequency, and the number of tachometer pulses per revolution.

number_of_tach_points = (*total_rows* \ *total_cols* – 1) \ *num_of_points* + *last_valid_index*

tach_pulses_per_rev = *abscissa_deltaX*

tach_frequency = *abscissa_firstX*

For each tachometer counter value in the tachometer data result (defined as tach_cnt[i]):

tach_time_sec[i] = tach_cnt[i] / tach_frequency

From each consecutive pair of tachometer points, you can compute and instantaneous RPM:

tach_pulse_delta_sec = tach_time_sec[i] – tach_time_sec[i – 1]

tach_pulses_per_min = 60 \ tach_pulses_per_sec

tach_rpm = tach_pulses_per_min / tach_pulses_per_rev

There may be a delay between the tachometer zero count value (when the tachometer started counting) and the first sample in the time capture data. From the SDF_VECTOR_HDR associated with the tachometer data result, you can locate the corresponding SDF_CHANNEL_HDR. The delay, in seconds, is in the *userDelay* field.

There is only one tachometer result in a time capture.

External Trigger Data

External trigger data is identified in a time capture file with the following SDF_DATA_HDR attributes:

dataType	48 (external trigger data)
ydata_type	LONG32
yIsComplex	0
scanData	0

If the trigger mode is set to be the external trigger at the time the capture was acquired and the tachometer is turned on, then this data result may be present.

Each point in the external trigger data is a 32 external trigger counter value which represents time. There is one time value associated with each scan of the time capture. The external trigger counter value is converted to a time value the same way as the tachometer data (see above).

Waterfall/Map/MultiSpectrum/Spectrogram Data

Many instrument create multiple scans of data and display them as a waterfall, a map, a multispectrum, or a spectrogram. In an SDF data file, this appears as multiple scans of a data result.

When this type of data is in the file, the *scanData* field in the SDF_DATA_HDR for a result is set to TRUE (1) and the SDF_FILE _HDR indicates that either an SDF_SCAN_STRUCT record or an SDF_SCAN_BIG record is in the file (or both). The *last_scan_index* field in either of these records indicates how many scans of data are in the file.

A scan variable may be:

- A time offset from when the measurement was started
- A scan offset from when the measurement was started
- The RPM when the scan was measured
- The average count when the scan was measured
- The overload when the scan was measured
- Information flags when the scan was measured

The first scan variable is in the SDF_SCAN_STRUCT record. The rest of the scan variables are in SDF_SCAN_VAR records. Most instruments support only one scan variable, which is in the SDF_SCAN _STRUCT record.

The HP 89410A and HP 89440A support 4 scan variables as follows:

Scan variable	Description
0 (SDF_SCAN_STRUCT)	Scan offset from when the measurement was started
1 (SDF_SCAN_VAR)	Same as above
2 (SDF_SCAN_VAR)	The average count when the scan was measured
3 (SDF_SCAN_VAR)	Information flags when the scan was measured (see below)

Information flags for scan variable 3:

Bit	Meaning when bit is set
0	row channel overload
1	column channel overload
8	questionable data
9	uncalibrated data
15	real-time data
16	digital demodulation sync search is on
17	digital demodulation pulse search is on
24	digital demodulation sync found (only valid if sync search is on)
25	digital demodulation pulse found (only valid if pulse search is on)

Example File

The following multi-page illustration shows the contents of a simple measurement file (of the SDF 1.) It defines just one trace. The trace's Y-axis values are contained in a vector of data at the end of the file (in the record SDF_YDATA_HDR). The trace's X-axis values are calculated from information in the record SDF_DATA_HDR.

File power.dat
B

SDF_FILE_HDR

recordType:	10
recordSize:	64
revisionNum:	0
applic:	1
yearStamp:	0
monthDayStamp:	0
hourMinStamp:	0
applicVer:	
num_of_DATA_HDR_record:	1
num_of_VECTOR_record:	1
num_of_CHANNEL_record:	1
num_of_UNIQUE_record:	0
num_of_SCAN_STRUCT_record:	0
num_of_XDATA_record:	0
offset_of_DATA_HDR_record:	168
offset_of_VECTOR_record:	1532
offset_of_CHANNEL_record:	1550
offset_of_UNIQUE_record:	−1
offset_of_SCAN_STRUCT_record:	−1
offset_of_XDATA_record:	−1
offset_of_YDATA_record:	1692

SDF_MEAS_HDR

recordType:	11
recordSize:	102
unique_record:	−1
centerFreq:	2.560000E+04
spanFreq:	6.553600E+04
blockSize:	1024
zoomModeOn:	0
startFreqIndex:	0

stopFreqIndex: 400
averageType: 1
averageNum: 10
pctOverlap: 0.000000E+00
measTitle:
videoBandWidth: 0.000000E+00

SDF_DATA_HDR 0
recordType: 12
recordSize: 114
unique_record: –1
dataTitle: Power Spectrum
domain: 0
dataType: 2
num_of_points: 513
last_valid_index: 512
abscissa_firstX: 0.000000E+00
abscissa_deltaX: 1.280000E+02
xResolution_type: 0
xdata_type: 3
xPerPoint: 0
ydata_type: 3
yPerPoint: 1
yIsComplex: 0
yIsNormalized: 0
yIsPowerData: 1
yIsValid: 1
first_VECTOR_recordNum: 0
total_rows: 1
total_cols: 1
unitLabel: Hz
unitFactor: 6.283190E+00
0
0
–2
0
0
0
0
2
yUnitValid: 0
unitLabel:
unitFactor: 0.000000E+00

0
0
0
0
0
0
0
0

SDF_VECTOR_HDR 0

recordType:	13
recordSize:	18
unique_record:	–1
the_CHANNEL_record[0]:	0
the_CHANNEL_record[1]:	–1
pwrOfChan[0]:	96
pwrOfChan[1]:	0

SDF_CHANNEL_HDR 0

recordType:	14
recordSize:	146
unique_record:	–1
channelLabel:	Chan 1
moduleId:	
serialNum:	P46.464646
windowType:	1
windowCorrMode:	0
windowBandWidth:	1.500000E+00
windowTimeConst:	0.000000E+00
windowTrunc:	0.000000E+00
wideBandCorr:	1.632993E+00
narrowBandCorr:	2.000000E+00
weight:	0
delay:	0.000000E+00
range:	2.000000E+00
direction:	3
pointNum:	1
coupling:	1
overloaded:	0
intLabel:	V
unitLabel:	g
unitFactor:	9.806650E+00

0
2
–4
0
0
0
0
0
int2engrUnit: 1.000000E-02
inputImpedance: 1.000000E+06

SDF_YDATA_HDR
recordType: 17
recordSize: 2058
7.688535E-03: 513 real points
1.922215E-03: |
9.141845E-10: V
.
.
.
2.377593E-12