Appendix A Area Detector Frame Formats

Bruker has used several different frame file formats over the years. These are:

- Bruker fixed-format/free-line frame image file.
- Bruker fixed-format frame image file.
- XENGEN frame image file.

By reading the first 240 characters of the file, you can identify the file format.

A-1 Uncompression algorithm for any Bruker frame file (excludes XENGEN).

- Read first 240 bytes of header.
    - Is this a Bruker frame file? The first 8 bytes will be "FORMAT :", bytes 81 to 88 will be "VERSION:", and bytes 161 to 168 will be "HDRBLKS:".
    - Which FORMAT? 86 or 100. This defines the image compression scheme used.
    - Which VERSION? 1 to 16. This defines which header items are present.
- Get header block size from HDRBLKS: The total size is 512 * hdrblks in bytes.
- Read entire header.
- Get image block size from NPIXELB. The total size is nbytes * nrows * ncolumns.
- Read image block.
- FORMAT 100:
    1. Get number of underflow entries from NOVERFL and size of underflow table from NPIXELB. The total size is nbytes * nentries padded to a multiple of 16.
    2. Read underflow table.
    3. Get number of 2-byte overflow entries from NOVERFL. The total size is 2 * nentries padded to a multiple of 16.
    4. Read 2-byte overflow table.
    5. Get number of 4-byte overflow entries from NOVERFL. The total size is 4 * nentries padded to a multiple of 16.
    6. Read 4-byte overflow table.
- FORMAT 86:
    1. Get number of overflow entries from NOVERFL. The total size is nentries * 16 padded to a multiple of 512 bytes.
    2. Read overflow table.
- Expand 1-byte or 2-byte image to a 4-byte image.
- FORMAT 100:
    1. Sequentially loop over all pixels
    2. 1-byte image: Replace any 255 pixel with next 2-byte overflow value (which may be 255 or 65535).
    3. 1-byte or 2-byte image: Replace any 65535 pixel with next 4-byte overflow value (which may be 65535).

    4. Handle underflow, baseline, or neither. (not applicable to multi-wire images).
        a. Replace any 0 pixel with next underflow value.
        b. If not -1, add baseline

- FORMAT 86:
    1. Sequentially loop over all overflows in overflow table
    2. Extract offset and pixel value pairs. Replace image offset with pixel value.
- Get the linear scaling parameters from frame header.
    - If 0.1 and 0.0, then covert to double: D[] = 0.1 * I[] + 0.0;
    - If not 1.0 and 0.0, then scale all pixels: I[] = A * I[] + B + 0.5.

Note: See FrmUtility sources for example of reading Bruker frame files in "C".

A-2 Bruker Floating Point Frame Format

In 2010, Bruker wanted to improve data quality. Performing all frame corrections in integer math was identified as a problem.

So, Bruker updated the frame file format to allow storing of floating point frame images. Actually, tenths of counts are now stored. Counts from minus baseline to 429496729.5 – baseline*10 can be stored. These frames are written by DIFFRAC.SUITE programs and can be identified by "LINEAR: 0.1 0.0" in the frame header. This frame file format is often called the BOOSTER format. [We are not concerned with floating point frames here.]

Additionally, new frame header items were added, but these are not related to the floating point image. Typically, frame file versions of 15, 16, and 17 are BOOSTER floating point frames, but do not rely on the version number to identify floating point frames.

A-3 Bruker Fixed-Format/Free-Line Frame Format

In 2000, Bruker wanted to improve compression of CCD frames. The original file compression scheme was optimized for multi-wire or gas detectors, where pixels started at zero counts. CCD frames can contain negative counts (after subtracting out the postscan and/or dark current), but a baseline is added to all pixels so that all pixels start at zero again. CCD frames rarely compressed to 8-byte frame files. Also, the original overflow table limited the frame size to maximum of 2048 x 2048.

So, Bruker updated the frame file format to improve the file compression scheme. As before, counts from minus baseline to 4294967295 – baseline can be stored. These frames are written by the BIS program and used by APEX1, PROTEUM2 and PILOT programs and can be identified by "FORMAT: 100" in the frame header. This frame file format is often called the BIS / PROTEUM format.

Additionally, new frame header items were added. Those new header items that are used for the improved file compression scheme are identified by "100" in the table below. Other new frame header items are identified by the version number. Typically, frame file versions of 11, 12, 13, and 14 are BIS / PROTEUM frames, but do not rely on the version number to identify this image compression scheme.

Bruker software will use the combination of the below compression options which produces the smallest file.

- Subtracting or not-subtracting the baseline.
- Compressing the 4-byte image to 1-byte with overflows as 255, 2-byte image with overflows as 65535, or leave as 4-byte image.
- No underflow table, 1- byte underflow table, or 2-byte underflow table.
- No overflow table, 1-byte overflow table, 2-byte overflow table, or both 1-byte and 2-byte overflow tables.

At the top level, a frame in Bruker format can be represented as:

Header
Compressed image data
Underflow data
Overflow data
Overflow data
Trailer

Header
The header information is essentially identical to the Fixed-Format Frame Format with these changes:

- FORMAT: is 100, instead of 86.
- HDRBLKS: The restriction that the number of header blocks is a multiple of 5 has been eliminated.
- The 72-character data for each header line is in free-format. All white space is spaces (no tabs, nulls, linefeeds, etc).

See **A-6 Bruker Frame File Header Items** for descriptions of the header information.

Image
The image data is a simple byte stream (no record delimiters) with one, two, or four bytes per pixel, as defined in the header. If the data are two bytes per pixel, the least-significant byte is first.

The order of the pixels in the file corresponds to the display of pixels on a screen, like the raster-scan order of a CRT display. The first pixel in the file corresponds to the upper-left corner of the display, the 512th to the upper right, and the last pixel to the lower right. This data ordering convention is the one used by most display devices and by the X-Windows library. Frames display as if viewed from the source toward the detector.

Underflow table
After subtracting the CCD baseline, any negative or zero pixels are stored in an underflow table. The number of underflow entries is stored as first value of the NOVERFL: header line. The byte size of the underflow table is stored as second value of NPIXELB header line.

The underflow table is stored in binary 1-byte or 2-byte format immediately following the compressed image data. The underflow block is padded with zeros to a multiple of 16-bytes.

Overflow table(s)
The number of 1-byte overflow entries is stored as the second value of the OVERFL: header line. The number of 2-byte overflow entries is stored as the third value of the OVERFL: header line.

Each overflow table is stored in binary 2-byte or 4-byte format immediately following the underflow table. Each overflow block is padded with zeros to a multiple of 16-bytes.

Trailers
The trailer has not changed. It was rarely used.

A-4 Bruker Fixed-Format Frame Format

The Bruker format addresses issues of data portability, speed of image I/O and display, and storage of all the data acquisition conditions required to reduce the data. Header and overflow information are stored in portable ASCII, rather than binary. Critical data, such as setting angles, can thus be stored portably as formatted floating-point values. The pixel values themselves are binary, but they are assured to start on a 512-byte boundary for quick, easy access (using Byte IO routines), and they are stored as a byte stream without delimiters of any kind. This format is completely transportable between all computers and networks, including PCs, NT workstations, and SGI systems. The only coding difference on these systems is that big-endian systems such as SGI must byte-swap the pixel values in the (relatively rare) case where it encounters a 16-bit frame.

Up to 32 bits/pixel are supported, although frames are stored where possible as 8 or 16 bits per pixel with an overflow table like that on earlier systems (but this overflow table is ASCII, as described below). BYTE ORDER OF PIXEL VALUES IN FRAMES WHICH ARE STORED IN THIS FORMAT ON DISK, TAPE, ETC. IS ALWAYS "LITTLE-ENDIAN" (LSB first, even though the header contains a flag indicating the pixel byte ordering). Bruker does not support big-endian frames that otherwise look like this format. The 16-bit format is fully implemented. If more than 4096 overflows occur on attempted compression to 8 bits, the frame buffer will compress to 16 bits with an overflow table (the SAXS HI-STAR and SMART CCD frame buffers accumulate 32 bits/pixel internally).

At the top level, a frame in Bruker format can be represented as:

Header
Image data
Overflow table
Trailer 1
.
.
.
Trailer N

Header
The header is 1) typable ASCII and 2) an even multiple of the 512-byte disk block size. Each entry in the header is an 80-byte unterminated ASCII line. Thus, the header size is always a multiple of 5 disk blocks to hold an integral number of lines. The first eight characters of each item contain the item name, with the remaining 72 characters for ASCII data. Thus, the organization of the header can be represented as:

HEADER
Item_name1 (8 char): Item_Data1 (72 chars)
Item_name2 (8 char): Item_Data2 (72 chars)
.
.
.
Item_nameN (8 char): Item_DataN (72 chars)

This organization has several advantages. First, fast random access can be performed on the header to pull out or rewrite an individual entry. During random access, the item names provide a method for checking for corruption of the header block.

Second, frames are actually TYPEable on both PCs and workstations if the screen wraps at 80 columns. The padding to a multiple of 5 disk blocks contains ASCII dot characters. The last characters of the padding are a CTRL+Z and a CTRL+D. This convention provides a "warning indicator" on a workstation during which a CTRL+S or CTRL+C can be issued, before the binary pixel values start to appear. On a PC, an MS-DOS TYPE command stops when it hits the CTRL+Z. Presence of the item names makes the output easily interpretable.

Third, the format is highly portable, because different byte and word ordering conventions don't enter into the interpretation of ASCII data. Items currently contained in the header are listed below in positional order. The format is expandable. New information to be added to the header will always be added at the end. A version number is present at the start of the header so programs can deal with new header information.See **A-6 Bruker Frame File Header Items** for descriptions of the header information.

Image Data
The image data is a simple byte stream (no record delimiters) with one, two, or four bytes per pixel, as defined in the header. If the data are two bytes per pixel, the least-significant byte is first.

The order of the pixels in the file corresponds to the display of pixels on a screen, like the raster-scan order of a CRT display. The first pixel in the file corresponds to the upper-left corner of the display, the 512th to the upper right, and the last pixel to the lower right. This data ordering convention is the one used by most display devices and by the X-Windows library. Frames display as if viewed from the source toward the detector.

Overflow Table
The overflow table is stored as ASCII values. Each overflow entry is 16 characters, comprising a 9-character

intensity and a 7-character pixel # offset. The table is padded to a multiple of 512 bytes. In an 8-bit frame, any pixel with a value of 255 in the image needs to be looked up in the overflow table to determine its true value (even if the true value is 255, to allow overflow table validity checks, which could not otherwise be made). In a similar manner, any pixel in a 16-bit frame with a value of 65535 must be looked up in the overflow table to determine its true value. To look up a pixel value, compute its pixel displacement (for example, in a 512x512 frame, 512*j + k, where j is the zero-based row number and k is the zero-based column number), and compare the displacement with that of each overflow table entry until a match is found. While the overflow table is normally sorted on displacement, it is not guaranteed to be sorted, so we recommend that you search the whole table until you find a match.

Trailers

The header contains a byte pointer (item name "TRAILER", set to zero if none) to the location in the file of any "trailer" data that has been tacked onto the image. This provision won't be used immediately, but provides later expansion capability. For example, if a frame is derived from processing some other (parent) frame, the header of the parent could be tacked onto the processed frame as a trailer to provide processing history. The format of a trailer is:

TRAILER

Trailer size in 512-byte blocks (1 byte)

3-byte pad

Trailer data

A-5 XENGEN Frame Format

Older Bruker software (sold when we were Siemens or Nicolet) had the ability to read and write frame files in the XENGEN frame format. This feature is now obsolete, although the ability to read XENGEN frames still exists in some Bruker software programs. If you encounter any old XENGEN frame files, try using the ToSAXII conversion utility.

A-6 Bruker Frame File Header Items

Mnemonic	Offset	Version/Format	Type	Description
FORMAT :	0	1	Int	Frame format. Always “86” or "100" for Bruker-format frames. 86 means fixed format header values and old frame compression format (see section: A-4). 100 means new free-format header values and image compression format (see section: A-3). Note: Says nothing about the file extension; *.sfrm files can be either FORMAT 86 or 100. Same with *.gfrm files.
VERSION:	1	1	Int	Header version #, such as: 1 to 17 (6 is obsolete). Defines what values are specified on the various header lines. Note: Versions 11 to 17 are typically only with FORMAT 100. Note: FORMAT: 100 VERSION: 1 were early beta software. These are actually version 11 headers. Typical versions from various software are: BIS: 1*(beta), 11, 12, 13, 14, 15. BOOSTER: 15, 16. (17 is reserved). FRAMBO: 1, 2, 3, 4, 5?, 8, 8+?, 8++?. GADDS: 3, 4, 5, 8, 8+, 8++. MICRODIF: 6 (completely obsoleted) See FrmUtility (or FrmFix) SAXS: 3?, 4, 5, 8, 8+, 8++. SMART: 7, 9, 9+, 9++, (10 for special project’s SMART?). Note: 8+, 8++, 9+, 9++ means another revision of this version. Header items were added, but version number stayed the same.

`HDRBLKS:`	2	1	Int	Header size in 512-byte blocks, such as 10 or 15. Determines where the image block begins.
`TYPE   :`	3	1	C	String indicating kind of data in the frame. Used to determine if a spatial correction table was applied to the frame image. Valid primary types are: BIS uses: FLOOD FIDUCIAL PLATE UNWARPED UNWARPED LINEAR Unknown Axis Scan Image Two-Theta Axis Scan Image Omega Axis Scan Image Phi Axis Scan Image Chi/Kappa Axis Scan Image Older Bruker software uses: DEFAULT Add frame Rotation frame SCAN FRAME UNWARPED UNWARPED LINEAR FLOOD FRAME FLOOD TABLE FIDUCIAL PLATE BACKGROUND <A> & <B> COMBINATION POLE FIGURE, [POLAR | STEREO | WULFF] STRESS, SCHEME Sometimes the primary type is appended with CALIBRATED (see DETPAR line) FIBER LPA MULT_FF PLATE SMOOTHED Pole figure type may be appended with: INTERPOLATED INVERTED NORMED ROTATED SCHEME SYMMETRIZED TILTED
`SITE   :`	4	1	C	Site name, from Edit > Configure > User settings or from Client program.
`MODEL  :`	5	1	C	Diffractometer model, from SAXI$ADMODEL or INI file. Typically this includes the goniometer type, the goniometer serial number, the stage, and the stage parameters. Examples:

				D8 [05-1234] with KAPPA[49.9876] GADDS-CST[05-1234] D8-TH-TH or D8-VERTICAL Can be used to convert angles from Eulerian space, back to native space (Theta-Theta or Kappa).
USER :	6	1	C	Username, from Edit > Configure > User settings of from Client program.
SAMPLE :	7	1	C	Sample ID, from Project > New/Copy/Edit or from Client program.
SETNAME:	8	1	C	Basic data set name
RUN :	9	1	Int	Run number within the data set, usually starts at 0, but 1 for APEX2.
SAMPNUM:	10	1	Int	Specimen number within the data set
TITLE :	11-18	1	C	User comments (8 lines)
NCOUNTS:	19	1,9,10	1: Int 9: 2 Int 10: 5 Int 11: 2 Int	Total frame counts (followed by reference counts in Ver 9). 9: Adds direct-beam monitor counts to NCOUNTS line. Default to 0 if missing. 10: Adds baseline offset [MultiWire: 0, CCD: 32 or 64], CCD experimental temp*100 C, and detector temp *100 C. 11: Removed the version 10 additions. Note: baseline offset also in NEXP line. Note: temps also on LOWTEMP line.
NOVERFL:	20	1	86: Int 100: 3Int	Number of overflows when compression frame. This has nothing to do with CCD chip overflows when collecting a frame. 86: Number of overflows. 100: Number of underflows (-1 = no baseline subtraction), Number of 1-byte overflows, Number of 2-byte overflows.
MINIMUM:	21	1	Int	Minimum counts in a pixel (uncompressed value)
MAXIMUM:	22	1	Int	Maximum counts in a pixel (uncompressed value)
NONTIME:	23	1	Int	Number of on-time events
NLATE :	24	1	Int	Number of late events. Always zero for many detectors.
FILENAM:	25	1	C	(Original) frame filename
CREATED:	26	1	C	Date and time of creation
CUMULAT:	27	1	R	Accumulated frame exposure time in seconds FRAMBO/GADDS/SAXS: Adds time for all exposures in this frame. CCD & BIS: Adds times for only correlation frames, excludes rescans and attenuated frame times.
ELAPSDR:	28	1	R	Requested time for last exposure in seconds FRAMBO/GADDS/SAXS: Typically same as CUMULAT: SMART: last image taken in frame (1x, nx, or ?x) BIS Classic: unknown BIS: last image taken in frame (1X, NX, MX, or AX)
		13	nR	BIS: AX, MX, NX, 1X images requested times in reverse order.
ELAPSDA:	29	1	R	Actual time for last exposure in seconds. Uses same definition of last exposure as in ELAPSDR:
		13	nR	BIS: AX, MX, NX, and 1X images actual times in reverse order.
OSCILLA:	30	1	Int	Nonzero if acquired by oscillation
NSTEPS :	31	1	Int	# steps or oscillations in this frame
RANGE :	32	1	R	Scan range in decimal degrees (unsigned)
START :	33	1	R	Starting scan angle value, decimal degrees
INCREME:	34	1	R	Scan angle increment between frames (signed)
NUMBER :	35	1	Int	Sequence number of this frame in series, usually starts at 0, but 1 for APEX2
NFRAMES:	36	1	Int	Total number of frames in the series

ANGLES :	37	1	4R	Diffractometer angles in Eulerian space (2T, OM, PH, CH). Must be converted from Kappa or theta-theta space to Eulerian space.
NOVER64:	38	1	Int	Number of pixels > 64K (actually LinearThreshold value)
			11: 3Int	11: Adds number of pixels > 64K after 8x rescan and number of pixels > 64K after 8x rescan with attenuator.
			15: nInt	15: 1x, nx, mx, and ax values for number of topped pixels.
NPIXELB:	39	1	Int	Number of bytes/pixel, such as 1, 2, or 4.
			100: 2Int	100: Adds number of bytes per pixel in under flow table (0, 1, or 2).
NROWS :	40	1	Int	Number of rasters in frame, such as 512, 1024, 2048, or 4096.
		16	16: 2Int	16: Adds number of panels in vertical direction. 1 for most detectors. Usually 2 for APEX-II / Pt135 detectors.
NCOLS :	41	1	Int	Number of pixels/raster, such as 512, 1024, 2048 or 4096
		16	16: 2Int	16: Adds number of panels in horizontal direction. 1 for most detectors. Usually 2 for APEX-II / Pt135 detectors.
WORDORD:	42	1	Int	Order of bytes in word (0=LSB first)
LONGORD:	43	1	Int	Order of words in a longword (0=LSW first)
TARGET :	44	1	C	X-ray target material: Cu, Mo, Ag, Fe, Cr, Co, Ni, W, Mn, or other.
SOURCEK:	45	1	R	X-ray source voltage in kV
SOURCEM:	46	1	R	X-ray source current in mA
FILTER :	47	1	C	Filter/monochromator setting: Such as: Parallel, graphite Ni Filter C Filter Zr Filter Cross coupled Goebel Mirrors
CELL :	48-49	1	6R	Unit cell A,B,C,ALPHA,BETA,GAMMA
MATRIX :	50-51	1	9R	Orientation matrix (P3 conventions)
LOWTEMP: …TEMP:	52	1,8,9,13	1: Int 8: Int,3R 9: 3Int 13: 3Int+	Low temp flag. Note: Version 8 and 9 headers diverge. 8: Adds sub-mnemonic TEMP: with set point K, ramp rate deg/min, hold time sec. Removed in Version 9. 9: Adds experimental temperature in hundredths C and CCD sensor temperature in hundredths C. 13: Optionally adds TEMP: R,R,Int with set temp in K, ramp rate in dpm, and hold time in seconds. If you do not have a TC, these values are not added. Note: Version 8 diverges from version 9, 10, 11, 12.
ZOOM :	53	1	3R	Zoom: Xc, Yc, Mag used for HI-STAR detectors: 0.5 0.5 1.0
CENTER :	54	1	2R 11: 4R	X, Y of direct beam at 2-theta = 0. These are raw center for raw frames and unwarped center for unwarped frames. 11: Xraw, Yraw, Xunw, Yunw.
DISTANC:	55	1	R	Sample-detector distance, cm (see CmToGrid value)
		11+,12	2R	Adds: Sample-detector grid/phosphor distance, cm
TRAILER:	56	1	Int	Byte pointer to trailer info
COMPRES:	57	1	C	Compression scheme ID, if any. Such as: NONE
LINEAR :	58	1	C	Linear scale, offset for pixel values, typically 1.0, 0.0. For BOOSTER frames, typically 0.1, 0.0 (intensity divided by ten). Note: A few early SMART frames mistakenly store the reciprocal value for the linear scale (so value was below 1.0).
PHD :	59	1	2R	Discriminator: Pulse height settings. X100 and X1000 only.
		17		17: Stores CCD phosphor efficiency (first field).
PREAMP :	60	1	R	Preamp gain setting. X100 and X1000 only.

		9	R	SMART 9: Stores Roper CCD gain table index value.
		11	2Int	BIS 11: Stores preamp gain and speed for CCD chip (as Ints)
			2R	BOOSTER: Back to floats.
CORRECT:	61	1	C	Flood table correction filename, UNKNOWN or LINEAR. Note: Some BIS versions place the path here too.
WARPFIL:	62	1	C	Brass plate correction filename, UNKNOWN or LINEAR. Note: Some BIS versions place the path here too. Note: A filename here does NOT mean that spatial correction was performed. See TYPE and string "UNWARP" to determine that.
WAVELEN:	63	1, 8+	1: 3R 8+: 4R 9: 3R 11: 4R	Version 1 to 7, 9, 10: Wavelengths (average, a1, a2) Version 8+: adds beta wavelength (removed in 9). Version 11: adds beta wavelength.
MAXXY :	64	2	2R	X,Y pixel # of maximum counts (from lower corner of 0,0) BIS: Has "X+1" (bug).
AXIS :	65	3	Int	Scan axis ib Eulerian space (1-4 for 2-theta, omega, phi, chi) (0 = none, 2 = default).
ENDING :	66	3	4R	Actual goniometer angles at end of frame in Eulerian space.
DETPAR :	67-68	4	6R	Detector position corrections (dX,dY,dDist,Pitch,Roll,Yaw) Note: GADDS and SAXS reused the Pitch variable for a 2T offset, whenever the TYPE line contained CALIBRATED.
LUT :	69	4	C	Recommended display lookup table
DISPLIM:	70	4	2R	Recommended display limits
PROGRAM:	71	4	C	Name and version of program writing frame, such as: SAXS for WNT V4.0.07 BIS V1.1.11 & BCP 1.1.10
ROTATE :	72	5	Int	Nonzero if acquired by rotation of phi during scan. Also if oscillates between phi limits.
BITMASK:	73	5	C	File name of active pixel mask associated with this frame or $NULL
OCTMASK:	74-75	5	8Int	Octagon mask parameters to use if BITMASK=$null. Min X, Min X+Y, Min Y, Max X-Y, Max X, Max X+Y, Max Y, Max Y-X.
ESDCELL:	76-77	7	6R	Unit cell parameter standard deviations
DETTYPE:	78	7, 8+	C	Detector or CCD chip type (as displayed on CEU). Default is MULTIWIRE.
			8+:C,2R	Version 8+: adds sub-mnemonics PIXPERCM: & CMTOGRID:. If these values are missing, the defaults are:

DETTYPE:	PIXPERCM:	CMTOGRID:
MULTIWIRE	47.5	2.0
CCD-PXL-2K	56.02	0.8
CCD-PXL-ARR	32.00	0.4
CCD-PXL-KAF1500	51.2	0.8
CCD-PXL-L6500N	32.00	1.5
CCD-PXL-L6500F	32.00	1.5
CCD-PXL-L6000	56.02	0.3 (or 56.3,0.8?)
CCD-PXL-KAF2	81.92	0.8
CCD-PXL-KAF	81.92	0.8
CCD-PXL-MSPD	81.92	0.8
CCD-PXL	81.92	0.8
CCD-LDI	83.333	0.8
CCD-MMX	Imported frames must ask for settings	
UNKNOWN	Imported frames must ask for settings	
OTHER	Imported frames must ask for settings	

Note: Newer detectors always have pixpercm and cmtogrid values in the frame header.

			9,10: C, 5R	Version 9,10,11: adds Pix512PerCM, CmtoGrid, 0.00, BrassHoleSpacingCm, BeWindowThicknessCm.
			11: C, 2R,I,2R,I	Version 11: adds Pix512PerCM, CmtoGrid, Circular (1=Yes), BrassHoleSpacingCm, BeWindowThicknessCm, AccuratelyTimed (1=Yes). Note: Version 8 diverges from version 9,10,11
NEXP :	79	7,9,10	7: Int 9: 2Int 10: 5Int	Number of exposures: 1=single, 2=correlated sum (followed by fixed bias ADU in Vers 9). 9: Adds per-exposure bias level in ADU to NEXP line 10: Adds baseline offset (normally 32), Adds Orientation (value of SAXI$INVERTCCD), Adds Overscan flag (1 if overscan used) Note: Baseline, if missing, should be defaulted to 0 for multiwires, 32 for most ccds, and 64 for 2K ccds. (see DETTYPE).
CCDPARM:	80	7	5R	CCD parameters: readnoise, e/ADU, e/photon, bias, full scale Full scale is defined differently by programs SMART & DIFFRAC.BIS: Full well of any image, default 64K. BIS: Potential full linear scale if rescan and attenuator used.
CHEM :	81	7	C	Chemical formula in CIFTAB string, such as “?”
MORPH :	82	7	C	Crystal morphology in CIFTAB string, such as “?”
CCOLOR :	83	7	C	Crystal color in CIFTAB string, such as “?”
CSIZE :	84	7	C	Crystal dimensions (3 ea) in CIFTAB string, such as “?”
DNSMET :	85	7	C	Density measurement method in CIFTAB string, such as “?”
DARK :	86	7	C	Name of dark current correction or NONE. Note: Some BIS versions also stored path here.
AUTORNG:	87	7	5R	Auto-ranging: gain, high-speed time, scale, offset, full linear scale Note: If full linear scale is zero, then CCDPARM full scale is the full linear scale (BIS frames).
ZEROADJ:	88	7	4R	Goniometer zero corrections (refined in least squares)
XTRANS :	89	7	3R	Crystal XYZ translations (refined in least squares)
HKL&XY :	90	8	5R	HKL and pixel XY for reciprocal space scan. GADDS only.
AXES2 :	91	8	4R	Diffractometer setting linear axes (4 ea). (X, Y, Z, Aux)
ENDING2:	92	8	4R	Actual goniometer linear axes @ end of frame. (X, Y, Z, Aux)
FILTER2:	93	11	2R 15: 4R	Monochromator 2-theta angle and monochromator roll angle. 15: Adds beam tilt angle and attenuator factor.
LEPTOS :	94	14	C	String for LEPTOS.
CFR:	95	12	2C 15: C	Only in 21CFRPart11 mode, writes the checksum for header and image. Ends with <Ctrl-Z> <Ctrl-D> characters.

Note: Items in version 6 headers are obsolete (MICRODIF). Use FrmUtility (or FRMFIX) to convert these frame headers to version 8.
Note: Version 5 & 8 items are mainly for GADDS
Note: Version 7 & 9 items are mainly for SMART
Note: Version 9, 10, 13, 15, & 16 items do not add additional lines to the header, but instead appends additional values to a previously existing line.
Note: Version 17 reuses item.