

MarCCD Header Documentataion

from C code in frame.h and types.h

Documentation updated by R. Doyle Thu May 29 18:57:00 CDT 2008

Documentation updated by M. Blum Tue Oct 11 11:49:20 CDT 2005

Description documents marccd v0.17.1

Summary of file structure:

```
|-- 1024 bytes TIFF HEADER -----|
|-- 3072 byte frame_header structure ---|
|-- nfast*nslow*depth byte image -----|
```

The full header, as written to the file, is a TIFF header. The initial 1024 bytes are a minimal TIFF header with a standard TIFF TAG pointing to the image data and a private TIFF TAG pointing to this header structure. As written by mmx/marccd, the frame_header structure always begins at byte 1024 and is 3072 bytes long making the full header 4096 bytes.

immediately following the header is the image - it is of arbitrary size defined

by the header fields nfast, nslow and depth. The total size is nfast * nslow * depth bytes.

The meanings of the data types should be self evident:

(example: UINT32 is an unsigned 32 bit integer)

The exact C language definition is machine dependent but these are the most common definitions on a 32bit architecture cpu.

```
#define UINT16    unsigned short
#define INT16     short
#define UINT32    unsigned int
#define INT32     int
```

Currently frames are always written as defined below:

```
origin=UPPER_LEFT
orientation=HFAST
view_direction=FROM_SOURCE
```

/* This number is written into the byte_order fields in the native byte order of the machine writing the file */

```
#define LITTLE_ENDIAN    1234
#define BIG_ENDIAN       4321
```

/* possible orientations of frame data (stored in orientation field) */

```
#define HFAST    0    /* Horizontal axis is fast */
#define VFAST    1    /* Vertical axis is fast */
```

/* possible origins of frame data (stored in origin field) */

```
#define UPPER_LEFT    0
#define LOWER_LEFT    1
#define UPPER_RIGHT   2
#define LOWER_RIGHT   3
```

/* possible view directions of frame data for

```

    the given orientation and origin (stored in view_direction field) */
#define FROM_SOURCE          0
#define TOWARD_SOURCE       1

#define MAXIMAGES 9
#define MAXSUBIMAGES 4096
#define MAXFRAMEDIMENSION 8192

typedef struct frame_header_type {
    /* File/header format parameters (256 bytes) */
    UINT32 header_type; /* flag for header type (can be used as
magic number) */
    char header_name[16]; /* header name (MARCCD) */
    UINT32 header_major_version; /* header_major_version (n.) */
    UINT32 header_minor_version; /* header_minor_version (.n) */
    UINT32 header_byte_order; /* BIG_ENDIAN (Motorola,MIPS);
LITTLE_ENDIAN (DEC, Intel) */
    UINT32 data_byte_order; /* BIG_ENDIAN (Motorola,MIPS);
LITTLE_ENDIAN (DEC, Intel) */
    UINT32 header_size; /* in bytes */
    UINT32 frame_type; /* flag for frame type */
    UINT32 magic_number; /* to be used as a flag - usually to
indicate new file */
    UINT32 compression_type; /* type of image compression */
    UINT32 compression1; /* compression parameter 1 */
    UINT32 compression2; /* compression parameter 2 */
    UINT32 compression3; /* compression parameter 3 */
    UINT32 compression4; /* compression parameter 4 */
    UINT32 compression5; /* compression parameter 4 */
    UINT32 compression6; /* compression parameter 4 */
    UINT32 nheaders; /* total number of headers */
    UINT32 nfast; /* number of pixels in one line */
    UINT32 nslow; /* number of lines in image */
    UINT32 depth; /* number of bytes per pixel */
    UINT32 record_length; /* number of pixels between successive rows
*/
    UINT32 signif_bits; /* true depth of data, in bits */
    UINT32 data_type; /* (signed,unsigned,float...) */
    UINT32 saturated_value; /* value marks pixel as saturated */
    UINT32 sequence; /* TRUE or FALSE */
    UINT32 nimages; /* total number of images - size of each
is nfast*(nslow/nimages) */
    UINT32 origin; /* corner of origin */
    UINT32 orientation; /* direction of fast axis */
    UINT32 view_direction; /* direction to view frame */
    UINT32 overflow_location; /* FOLLOWING_HEADER, FOLLOWING_DATA */
    UINT32 over_8_bits; /* # of pixels with counts > 255 */
    UINT32 over_16_bits; /* # of pixels with count > 65535 */
    UINT32 multiplexed; /* multiplex flag */
    UINT32 nfastimages; /* # of images in fast direction */
    UINT32 nslowimages; /* # of images in slow direction */
    darkcurrent_applied; /* flags correction has been applied -
hold magic number ? */
    UINT32 bias_applied; /* flags correction has been applied -
hold magic number ? */
    UINT32 flatfield_applied; /* flags correction has been applied -
hold magic number ? */

```

```

        UINT32      distortion_applied; /* flags correction has been applied -
hold magic number ? */
        UINT32      original_header_type; /* Header/frame type from file that
frame is read from */
        UINT32      file_saved;          /* Flag that file has been saved, should
be zeroed if modified */
        UINT32      n_valid_pixels;     /* Number of pixels holding valid data -
first N pixels */
        UINT32      defectmap_applied; /* flags correction has been applied -
hold magic number ? */
        UINT32      subimage_nfast;     /* when divided into subimages (eg.
frameshifted) */
        UINT32      subimage_nslow;     /* when divided into subimages (eg.
frameshifted) */
        UINT32      subimage_origin_fast; /* when divided into subimages (eg.
frameshifted) */
        UINT32      subimage_origin_slow; /* when divided into subimages (eg.
frameshifted) */
        UINT32      readout_pattern;    /* BIT Code - 1 = A, 2 = B, 4 = C, 8 =
D */
        UINT32      saturation_level;   /* at this value and above, data
are not reliable */
        UINT32      orientation_code;   /* Describes how this frame needs
to be rotated to make it "right" */
        UINT32      frameshift_multiplexed; /* frameshift multiplex flag */
        UINT32      prescan_nfast;      /* Number of non-image pixels
preceding imaging pixels - fast direction */
        UINT32      prescan_nslow;      /* Number of non-image pixels
preceding imaging pixels - slow direction */
        UINT32      postscan_nfast;     /* Number of non-image pixels
following imaging pixels - fast direction */
        UINT32      postscan_nslow;     /* Number of non-image pixels
following imaging pixels - slow direction */
        UINT32      prepost_trimmed;    /* trimmed==1 means pre and post
scan pixels have been removed */
        char reserve1[(64-55)*sizeof(INT32)-16];

        /* Data statistics (128) */
        UINT32      total_counts[2];    /* 64 bit integer range = 1.85E19*/
        UINT32      special_counts1[2];
        UINT32      special_counts2[2];
        UINT32      min;
        UINT32      max;
        INT32       mean;                /* mean * 1000 */
        UINT32      rms;                 /* rms * 1000 */
        UINT32      n_zeros;             /* number of pixels with 0 value -
not included in stats in unsigned data */
        UINT32      n_saturated;         /* number of pixels with saturated
value - not included in stats */
        UINT32      stats_uptodate;     /* Flag that stats OK - ie data not
changed since last calculation */
        UINT32      pixel_noise[MAXIMAGES]; /* 1000*base noise value
(ADUs) */
        char reserve2[(32-13-MAXIMAGES)*sizeof(INT32)];

#if 0
        /* More statistics (256) */
        UINT16 percentile[128];

```

```

#else
/* Sample Changer info */
char          barcode[16];
UINT32       barcode_angle;
UINT32       barcode_status;
/* Pad to 256 bytes */
char reserve2a[(64-6)*sizeof(INT32)];
#endif

/* Goniostat parameters (128 bytes) */
INT32 xtal_to_detector;      /* 1000*distance in millimeters */
INT32 beam_x;                /* 1000*x beam position (pixels) */
INT32 beam_y;                /* 1000*y beam position (pixels) */
INT32 integration_time;     /* integration time in milliseconds */
INT32 exposure_time;        /* exposure time in milliseconds */
INT32 readout_time;         /* readout time in milliseconds */
INT32 nreads;               /* number of readouts to get this image */
INT32 start_twtheta;        /* 1000*two_theta angle */
INT32 start_omega;          /* 1000*omega angle */
INT32 start_chi;            /* 1000*chi angle */
INT32 start_kappa;          /* 1000*kappa angle */
INT32 start_phi;           /* 1000*phi angle */
INT32 start_delta;         /* 1000*delta angle */
INT32 start_gamma;         /* 1000*gamma angle */
INT32 start_xtal_to_detector; /* 1000*distance in mm (dist in um)*/
INT32 end_twtheta;          /* 1000*two_theta angle */
INT32 end_omega;            /* 1000*omega angle */
INT32 end_chi;              /* 1000*chi angle */
INT32 end_kappa;            /* 1000*kappa angle */
INT32 end_phi;              /* 1000*phi angle */
INT32 end_delta;            /* 1000*delta angle */
INT32 end_gamma;            /* 1000*gamma angle */
INT32 end_xtal_to_detector; /* 1000*distance in mm (dist in um)*/
INT32 rotation_axis;        /* active rotation axis (index into above ie.
0=twtheta,1=omega...) */
INT32 rotation_range;       /* 1000*rotation angle */
INT32 detector_rotx;         /* 1000*rotation of detector around X */
INT32 detector_roty;         /* 1000*rotation of detector around Y */
INT32 detector_rotz;         /* 1000*rotation of detector around Z */
INT32 total_dose;            /* Hz-sec (counts) integrated over full
exposure */
char reserve3[(32-29)*sizeof(INT32)]; /* Pad Gonisotat parameters to 128
bytes */

/* Detector parameters (128 bytes) */
INT32 detector_type;         /* detector type */
INT32 pixelsize_x;           /* pixel size (nanometers) */
INT32 pixelsize_y;           /* pixel size (nanometers) */
INT32 mean_bias;             /* 1000*mean bias value */
INT32 photons_per_100adu;    /* photons / 100 ADUs */
INT32 measured_bias[MAXIMAGES]; /* 1000*mean bias value for each
image*/
INT32 measured_temperature[MAXIMAGES]; /* Temperature of each detector
in milliKelvins */
INT32 measured_pressure[MAXIMAGES]; /* Pressure of each chamber in
microTorr */

```

```

/* Retired reserve4 when MAXIMAGES set to 9 from 16 and two fields removed,
and temp and pressure added
char reserve4[(32-(5+3*MAXIMAGES))*sizeof(INT32)];
*/

/* X-ray source and optics parameters (128 bytes) */
/* X-ray source parameters (14*4 bytes) */
INT32 source_type;          /* (code) - target, synch. etc */
INT32 source_dx;           /* Optics param. - (size microns) */
INT32 source_dy;           /* Optics param. - (size microns) */
INT32 source_wavelength;   /* wavelength (femtoMeters) */
INT32 source_power;        /* (Watts) */
INT32 source_voltage;      /* (Volts) */
INT32 source_current;      /* (microAmps) */
INT32 source_bias;         /* (Volts) */
INT32 source_polarization_x; /* () */
INT32 source_polarization_y; /* () */
INT32 source_intensity_0;  /* (arbitrary units) */
INT32 source_intensity_1;  /* (arbitrary units) */
char reserve_source[2*sizeof(INT32)];

/* X-ray optics parameters (8*4 bytes) */
INT32 optics_type;         /* Optics type (code)*/
INT32 optics_dx;           /* Optics param. - (size microns) */
INT32 optics_dy;           /* Optics param. - (size microns) */
INT32 optics_wavelength;   /* Optics param. - (size microns) */
INT32 optics_dispersion;   /* Optics param. - (*10E6) */
INT32 optics_crossfire_x;  /* Optics param. - (microRadians) */
INT32 optics_crossfire_y;  /* Optics param. - (microRadians) */
INT32 optics_angle;        /* Optics param. - (monoch. 2theta -
microradians) */
INT32 optics_polarization_x; /* () */
INT32 optics_polarization_y; /* () */
char reserve_optics[4*sizeof(INT32)];

char reserve5[((32-28)*sizeof(INT32))]; /* Pad X-ray parameters to 128
bytes */

/* File parameters (1024 bytes) */
char filetype[128];        /* Title */
char filepath[128];       /* path name for data file */
char filename[64];        /* name of data file */
char acquire_timestamp[32]; /* date and time of acquisition */
char header_timestamp[32]; /* date and time of header update */
char save_timestamp[32];   /* date and time file saved */
char file_comment[512];    /* comments - can be used as desired */
char reserve6[1024-(128+128+64+(3*32)+512)]; /* Pad File parameters to
1024 bytes */

/* Dataset parameters (512 bytes) */
char dataset_comment[512]; /* comments - can be used as desired */

/* Reserved for user definable data - will not be used by Mar! */
char user_data[512];

/* char pad[----] USED UP! */ /* pad out to 3072 bytes */

} frame_header;

```