NAME

bsfile – file format for Hawaii Mapping Research Group BS software

SYNOPSIS

#include <local/bs.h>

DESCRIPTION

A BS file is used for the storage of binary ping data operated upon by the Hawaii Mapping Research Group's BS processing system. The file is portable among machines of varying architecture by virtue of its XDR implementation. It is composed of a file header which describes the number of pings contained in the file followed by the pings themselves. Each ping contains a header structure followed by a stream of sensor (i.e., towfish compass, depth, pitch and roll), bathymetry and sidescan data of arbitrary length.

This file format was originally known as MR1 format due to its initial purpose of supporting the HAWAII-MR1 sonar. The format was renamed as BS in 2005 to indicate its more general support for bathymetry and sidescan data from various instruments, as well as to differentiate it from the raw HAWAII-MR1 acquisition format which is radically different. Any perceived comment on its inherent quality is wholly unintentional.

Following is a partial listing of the contents of the file bs.h which define the file and ping header structures and a few relevant constants:

```
/* version number, guaranteed to be strictly
   increasing and in chronological order */
#define MR1_VERSION_1_0         (6666)      /* obsolete version */
#define MR1_VERSION_2_0         (6667)      /* obsolete version */
#define BS_VERSION_1_0          (6668)      /* obsolete as of 2007/06/28 */
#define BS_VERSION_1_1          (6669)      /* obsolete as of 2007/11/27 */
#define BS_VERSION_1_2          (6670)      /* obsolete as of 2008/04/14 */
#define BS_VERSION_1_3          (6671)      /* obsolete as of 2010/03/10 */
#define BS_VERSION_1_4          (6672)      /* current version */

/* file flag bits */
#define BS_CLEAR                (0x0)
#define BS_SSSLANTRNG           (0x1)       /* sidescan are slant range */
#define BS_MSCPINGDELRST        (0x2)       /* ping delete/restore via mosaic GUI */
#define BS_MSCNAVEDIT           (0x4)       /* navigation edits via mosaic GUI */
#define BS_MSCBRKFILE           (0x8)       /* file break via mosaic GUI */
#define BS_MSCEDGETRIM          (0x10)      /* edge trims via mosaic GUI */

/* acquisition instruments */
#define BS_INST_UNDEFINED       (-1)
#define BS_INST_MR1             (0)
#define BS_INST_SEAMAPB         (1)
#define BS_INST_IMI30           (2)
#define BS_INST_IMI12           (3)
#define BS_INST_DSL120A         (4)
#define BS_INST_SEAMAPC         (100)
#define BS_INST_SCAMP           (150)
#define BS_INST_EM120           (2000)
#define BS_INST_EM1002          (2001)
#define BS_INST_EM300           (2002)
#define BS_INST_EM3000          (2003)
#define BS_INST_EM3002          (2004)
#define BS_INST_EM3000D         (2005)
#define BS_INST_EM3002D         (2006)
#define BS_INST_EM2000          (2007)
```

```
#define BS_INST_EM122           (2008)
#define BS_INST_EM302           (2009)
#define BS_INST_EM710           (2010)
#define BS_INST_SM2000          (2050)
#define BS_INST_RESON8101       (3000)
#define BS_INST_RESON8111       (3001)
#define BS_INST_RESON8124       (3002)
#define BS_INST_RESON8125       (3003)
#define BS_INST_RESON8150       (3004)
#define BS_INST_RESON8160       (3005)
#define BS_INST_AMS120          (4000)
#define BS_INST_REMUS           (4100)
#define BS_INST_KLEIN5000       (5000)
#define BS_INST_SEABEAM2000     (6000)
#define BS_INST_SEABEAM2100     (6010)
#define BS_INST_SEABEAM3012     (6050)
#define BS_INST_SSI             (7000)
#define BS_INST_SAICLLS         (8000)
#define BS_INST_EDGETECHSS      (9000)
#define BS_INST_EDGETECHSSM             (9001)
#define BS_INST_EDGETECHSSH             (9002)
#define BS_INST_EDGETECHSB      (9003)

/* source file formats */
#define BS_SFMT_UNDEFINED       (-1)
#define BS_SFMT_MR1             (0)
#define BS_SFMT_TTS             (1)
#define BS_SFMT_GSF             (1000)
#define BS_SFMT_GSFDUAL         (1001)
#define BS_SFMT_XTF             (1100)
#define BS_SFMT_SIMRADEM        (2000)
#define BS_SFMT_SIMRADMPB       (2001)
#define BS_SFMT_OIC             (4000)
#define BS_SFMT_OICLLS          (4001)
#define BS_SFMT_MSTIFF          (4100)
#define BS_SFMT_SIOSB2000       (6000)
#define BS_SFMT_SSIV21          (7000)
#define BS_SFMT_XSE             (8000)
#define BS_SFMT_JSF             (9000)

/* data type mask bits */
#define BS_DTM_NONE             (0)
#define BS_DTM_COMPASS                  (0x1)
#define BS_DTM_DEPTH                    (0x2)
#define BS_DTM_PITCH            (0x4)
#define BS_DTM_ROLL             (0x8)
#define BS_DTM_BATHYMETRY       (0x10)
#define BS_DTM_SIDESCAN                 (0x20)

typedef struct bsf_struct {
        int bsf_version;                /* file format version number */
        int bsf_count;                  /* number of objects */
        unsigned int bsf_flags;         /* BS_SSSLANTRNG, etc. */
        int bsf_inst;                   /* acquisition instrument */
```

```
        int bsf_srcformat;      /* source file format */
        char *bsf_srcfilenm;            /* source file name */
        char *bsf_log;                  /* processing log */
} BSFile;

typedef struct sns_struct {
        float sns_int;                  /* sample interval (secs) */
        int sns_nsamps;                 /* number of samples */
        float sns_repval;               /* single representative value of the sensor
                                           for an entire ping, usually derived from
                                           the full set of samples for that ping */
} Sensor;

typedef struct ps_struct {
        float ps_xmitpwr;               /* transmitter power (1=full) */
        float ps_gain;                  /* gain setting */
        float ps_pulse;                 /* pulse length (millisecs) */
        float ps_bdrange;               /* bottom detect range (m) */
        int ps_btycount;                /* number of bathymetry samples */
        int ps_btypad;                  /* number of trailing bathymetry pad samples */
        float ps_ssxoffset;                     /* across-track distance (m) or, for
                                           BS_SSSLANTRNG files, time (s) to first
                                           sidescan sample */
        int ps_sscount;                 /* number of sidescan samples */
        int ps_sspad;                   /* number of trailing sidescan pad samples */
        float ps_ssndrmask;             /* across-track distance to outer edge
                                           of nadir region data to be masked */
        float ps_ssyoffset;                     /* sidescan along-track offset (m) */
} PingSide;

#define PNG_CLEAR                       (0x0)
#define PNG_XYZ                         (0x1)   /* bathymetry is x/y/z instead
                                                   of x/z only */
#define PNG_ABI                         (0x2)   /* auxiliary beam info present */
#define PNG_BTYSSFLAGSABSENT            (0x4)   /* indicates that input file does
                                                   not contain bathymetry or
                                                   sidescan flags, i.e., the file
                                                   is in an older flagless format
                                                   version; all output files are
                                                   written with flags and this bit
                                                   is always unset when written
                                                   to output */
#define PNG_HIDE                        (0x8)   /* ping should not be displayed */
#define PNG_LOWQUALITY                  (0x10)  /* ping is of unacceptably low quality */
#define PNG_MSCHIDE                     (0x20)  /* ping should not be displayed
                                                   in a mosaic */
/* sidescan along-track offset mode */
#define PNG_SSYOM_UNKNOWN               (0)     /* unknown (all pre-BS-1.4 files) */
#define PNG_SSYOM_CONSTANT              (1)     /* constant offset for entire ping */
#define PNG_SSYOM_USEBTYY (2)           /* use bathymetry y-offsets */

#define PNG_BYTEALIGNSZ                 (8)     /* byte alignment constraint for
                                                   beginning of auxiliary beam
                                                   info section of data buffer */
```

```
typedef struct png_struct {
        unsigned int png_flags;         /* PNG_XYZ, etc. */
        struct timeval png_tm;          /* timestamp */
        float png_period;               /* ping period (secs) */
        double png_slon;                /* ship longitude (deg) */
        double png_slat;                /* ship latitude (deg) */
        float png_scourse;              /* ship course (deg) */
        float png_laybackrng;           /* towfish layback range (m) */
        float png_laybackbrg;           /* towfish layback bearing (deg, where 0=shipaxis,
                                           pos=port, neg=starboard) */
        double png_tlon;                /* towfish longitude (deg) */
        double png_tlat;                /* towfish latitude (deg) */
        float png_tcourse;          /* towfish course (deg) */
        Sensor png_compass;             /* towfish compass heading (deg, where 0=N, 90=E)
                                           with no magnetic correction applied to either
                                           the representative value or the sample array */
        Sensor png_depth;               /* towfish depth (m) */
        Sensor png_pitch;               /* towfish pitch (deg, where + is nose up) */
        Sensor png_roll;                /* towfish roll (deg, where + is port down) */
        int png_snspad;                 /* number of invalid trailing pad sensor samples */
        float png_temp;                 /* water temperature (deg C) */
        float png_ssincr;               /* sidescan increment in across-track distance (m)
                                           or, for BS_SSSLANTRNG files, time (s) */
        int png_ssyoffsetmode;          /* sidescan along-track offset mode */
        float png_alt;                  /* towfish altitude (m) */
        float png_magcorr;              /* magnetic correction (deg) */
        float png_sndvel;               /* sound velocity (m/sec) */
        float png_cond;                 /* conductivity (siemens/m) */
        float png_magx;                 /* magnetic field x (microteslas) */
        float png_magy;                 /* magnetic field y (microteslas) */
        float png_magz;                 /* magnetic field z (microteslas) */
        PingSide png_sides[ACP_NSIDES];
} Ping;

/* bathymetry per-sample flag bits
   (must fit in a 4-byte integer) */
#define BTYD_CLEAR              (0x0)   /* no flag -- sample is valid */
#define BTYD_MISC               (0x1)   /* invalidated for non-specific reason */
#define BTYD_EXTERNAL           (0x2)   /* invalidated by external (non-HMRG) software */
#define BTYD_MINMAXCLIP         (0x4)   /* deleted by min/max depth clip */
#define BTYD_MAXANGLE           (0x8)   /* exceeds maximum angle */
#define BTYD_MINANGLE           (0x10)  /* less than minimum angle */
#define BTYD_SWEDGE             (0x20)  /* deleted from edge of swath */
#define BTYD_SWRECT             (0x40)  /* deleted swath rectangle interior */
#define BTYD_MFSWAPERR          (0x80)  /* mapping function swap error */
#define BTYD_SRFABOVECLIP       (0x100) /* clipped from above a surface */
#define BTYD_SRFBELOWCLIP       (0x200) /* clipped from below a surface */
#define BTYD_XZPRECT                    (0x400)  /* deleted xz-profile rectangle interior */

/* sidescan per-sample flag bits (must
   fit in a 1-byte unsigned char) */
#define SSD_CLEAR               (0x0)   /* no flag -- sample is valid */
#define SSD_MISC                (0x1)   /* invalidated for non-specific reason */
#define SSD_EXTERNAL(0x2)       /* invalidated by external (non-HMRG) software */
```

```
#define SSD_MAXANGLE        (0x4)   /* exceeds maximum angle */
#define SSD_MINANGLE        (0x8)   /* less than minimum angle */
#define SSD_SWEDGE          (0x10)  /* deleted from edge of swath */
#define SSD_SWRECT          (0x20)  /* deleted swath rectangle interior */

/* AuxBeamInfo --
  This structure contains various bits of per-beam information
  necessary to reconvert back to a source multibeam format. */

/* auxiliary beam information flag bits */
#define ABI_CLEAR           (0x0)
#define ABI_SSVALID         (0x1)   /* sidescan valid */

typedef struct abi_struct {
        unsigned int abi_flags;         /* ABI_SSVALID, etc. */
        int abi_id;                     /* beam number */
        float abi_ssat0;                /* across-track distance of first sidescan sample */
        float abi_ssat1;                /* across-track distance of last sidescan sample */
} AuxBeamInfo;

typedef struct pd_struct {
        float *pd_compass;
        float *pd_depth;
        float *pd_pitch;
        float *pd_roll;
        float *pd_bty[ACP_NSIDES];
        unsigned int *pd_btyflags[ACP_NSIDES];
        float *pd_ss[ACP_NSIDES];
        unsigned char *pd_ssflags[ACP_NSIDES];
        AuxBeamInfo *pd_abi[ACP_NSIDES];
} PingData;
```

The first data object in the file is a *BSFile* structure. The structure is stored in the file by storing one XDR primitive for each of the structure's fields. These are stored in the same order as the fields are defined within the structure, i.e., an initial 4-byte XDR integer containing the file format version number, a second 4-byte XDR integer representing the count of pings contained within the file, a 4-byte XDR unsigned integer containing various flag bits, another 4-byte XDR integer representing the type of acquisition instrument, if any, which originally generated the data which is either in the file or from which the file's data were derived, another 4-byte XDR integer describing the source format of the data, if any, from which the file's data were derived, a string which is the name of the file containing the data, if any, from which the file's data were derived and finally a string containing a log entry. (Note that strings are stored as a 4-byte XDR integer describing the string's length followed by the bytes of the string, if any. No null terminator byte is stored in the file.)

The remainder of the file consists of a sequence of 0 or more pings, where each ping consists of some ping header information immediately followed by that ping's data samples. The ping header data is stored by storing one XDR primitive for each of the *Ping* structure's fields. These primitives are written in the order in which they are defined within the *Ping* structure, with the exception of the *png_snspad* field which is not stored in the file (as no sensor sample padding is ever written to a file and hence the value of this field is assumed to be 0). Wherever a field is of non-primitive type, i.e., is not a simple *int, unsigned int, float, double*, etc., but is rather a structure of some other type (i.e., *struct timeval, Sensor* or *PingSide*), then that substructure is stored by storing one XDR primitive for each of its fields in turn with certain exceptions as follows: (i) the fields of the *png_tm* substructure are stored as 4-byte XDR integers regardless of whether the host platform is 32- or 64-bit, and (ii) the *ps_btypad* and *ps_sspad* fields of the *PingSide* structure are not stored (as no bathymetry or sidescan sample padding is ever written to a file and hence the values of these fields are assumed to be 0). Note that the last member of the *Ping* structure is an array of two *PingSide*

structure elements. The port *PingSide* structure is stored first and the starboard *PingSide* structure is stored second.

Immediately after each ping's header its data samples are stored in the following order: (i) towfish compass, (ii) towfish depth, (iii) towfish pitch, (iv) towfish roll, (v) port bathymetry, (vi) port bathymetry flags, (vii) port sidescan, (viii) port sidescan flags, (ix) starboard bathymetry, (x) starboard bathymetry flags, (xi) starboard sidescan, (xii) starboard sidescan flags, (xiii) port auxiliary beam information, and (xiv) starboard auxiliary beam information.

The numbers of towfish compass, depth, pitch and roll samples are contained in the *sns_nsamps* fields of the *png_compass*, *png_depth*, *png_pitch* and *png_roll Sensor* structures embedded within the *Ping* structure. A single 4-byte XDR float is stored for each sample from these sensors.

The numbers of bathymetry samples for the two sides are contained in the *ps_btycount* fields of the port and starboard *PingSide* structures embedded within the *Ping* structure. Each bathymetry sample consists of either two or three consecutive 4-byte XDR float values depending upon the value of the *Ping* structure's *png_flags* field. If the **PNG_XYZ** bit of the latter is set then each sample is an x/y/z triplet with the first value representing across-track distance, the second value representing along-track distance and the third value representing depth. If the **PNG_XYZ** bit is not set then the data are in x/z format, i.e., the along-track distance is not present. The samples of each side must be sorted in order of increasing x, i.e., increasing across-track distance. Note that x will nearly always be non-negative for samples on both sides, e.g., an x-value of 5 indicates that a port sample lies 5 meters to the port side of nadir or that a starboard sample lies 5 meters to the starboard side of nadir. Negative x-values are rare though legal, and indicate that a sample actually lies on the opposite side of nadir, e.g. a port sample with an x-value of -5 lies 5 meters to the starboard side of nadir.

A sequence of bathymetry flag values follows the bathymetry samples, one flag value per sample where each flag value is stored as a 4-byte XDR unsigned integer. Each flag value is a composite bitmask of the bit-fields **BTYD_MISC, BTYD_DEPTHCLIP**, etc., and indicates whether a particular sample has been flagged as invalid for one or more reasons. A flag value of **BTYD_CLEAR** indicates that a sample has not been flagged as invalid.

The numbers of sidescan samples for the two sides are contained in the *ps_sscount* fields of the port and starboard *PingSide* structures embedded within the *Ping* structure. Each sidescan sample consists of a single 4-byte XDR float value representing a reflection intensity. Samples are stored in order of their increasing distance from swath nadir or, if the **BS_SSSLANTRNG** bit of the *bsf_flags* field of the *BSFile* structure is set, in order of their increasing time from the start of the ping.

A sequence of sidescan flag values follows the sidescan samples, one flag value per sample where the complete set of flag values for each side of a ping is stored as a single XDR byte array (via **xdr_bytes()**) containing a 1-byte flag value per sidescan sample. Each flag value is a composite bitmask of the bit-fields **SSD_MISC, SSD_MAXANGLE**, etc., and indicates whether a particular sample has been flagged as invalid for one or more reasons. A flag value of **SSD_CLEAR** indicates that a sample has not been flagged as invalid.

The final component of the ping data samples is the optional auxiliary beam information. If this component is present, a single *AuxBeamInfo* structure will be stored for each bathymetry sample. Those structures corresponding to the port data are stored first, followed by those structures corresponding to the starboard data. Each such structure is stored by storing one XDR primitive for each of its fields, written in the order in which they are defined within the *AuxBeamInfo* structure.

Note that the ordering of bathymetry samples, bathymetry flags and auxiliary beam information structures, which are all in one-to-one correspondence, is identical, e.g., the *n*th port bathymetry sample corresponds to the *n*th port bathymetry flag value and the *n*th port auxiliary beam information structure.

Although the sample padding described in the **bs**(3) manual page may be present within an in-memory image of a BS data set, such padding will never be stored into a BS disk file. Consequently, the various *png_snspad, ps_btypad* and *ps_sspad* fields are never written to BS files and are assumed to be 0 as noted above.

IEEE NaN (not-a-number) is used by this format to indicate that the value of a particular header field (e.g., towfish altitude) or data sample is unknown. It is common for most of the sensor samples (e.g., roll) in files derived from HAWAII-MR1, for instance, to be stored as NaNs because that instrument logs such samples at an irregular rate. Those samples are then transformed into a set of regularly logged samples by filling the gaps between the real samples with NaN placeholder values, with the expectation that in some later processing stage those placeholder values will be turned into real numbers via some form of interpolation.

I/O routines which read and write the structures described here are available from the **bs**(3) library.

FILES

/usr/include/local/bs.h

SEE ALSO

bs(3), **bs2asc**(1)

AUTHOR

Roger Davis, July 2005.