EVT File Structure for Etna, K2, Makalu, Mt. Whitney, QDR, and Rock Digitizers

The Kinemetrics EVT file consists of three structures, a TAG, a FILE HEADER, a FRAME HEADER and a DATA FRAME. The TAG precedes the FILE HEADER and each of the FRAME HEADERs.

Definition - *filename*.EVT structure

```
TAG             16 bytes
FILE HEADER     2040 or 2736     //  12 channel = 2040 bytes
                                 //  18 channel = 2736 bytes
```

------------ **for each 1/10 of a second of data**

```
TAG             16 bytes
FRAME HEADER    32 bytes
DATA            multiple of 3 bytes
     Minimum  1 channel    10 sps     1 *  1 * 3 =      3 bytes
     Maximum 18 channels 2000 sps   200 * 18 * 3 = 10,800 bytes
```

Data is in scans, lowest recorded channel first

These three structures repeat for all of the data recorded in the event.

Definition - TAG structure

```
/*------------------------------------------------------------*/
/*    TAG – Preceeds each File Header and Data Frame Header */
/*------------------------------------------------------------*/
/* STRUCTURE TYPE CODES */
#define KFF_K2_HDR 1     /* K2 header */
#define KFF_K2_FRAME 2   /* K2 FRAME; frame data follows */
#define KFF_BYTE_ORDER 1 /* 0=INTEL (LSB first) ,
                            1 = MOTOROLA (MSB first) */
#define KFF_VERSION 1
#define KFF_SYNC_CHAR 'K'

/* KFF_TAG. Separates and identifies structures */
Hex
Add
      typedef struct KFF_TAG {
000   unsigned char sync; /* sync character 'K', 0x4B */
001   unsigned char byteOrder; /* = 0 for LSB first (INTEL),
                                 1 for MSB first (MOTOROLA) */
002   unsigned char version; /* File format version; KFF_VERSION */
003   unsigned char instrumentType; /* instrument type code */
             /*   9, 0x09 = K2 */
              /* 10, 0x10 = Makalu */
              /* 20, 0x14 = New Etna */
              /* 30, 0x1E = Rock */
              /* 40, 0x28 = SSA2EVT */
004   unsigned long type; /* structure type code */
          /*  1 = File Header,  2 = Data Frame
008   unsigned short length; /* structure size in bytes */
Hex
Add
         /* # bytes in file header or in frame header */
00a   unsigned short dataLength; /* # of data bytes following the structure */
         /* 00 if file header, else # of data multiplexed data bytes */
00c   unsigned short id; /* unique instrument id (e.g. serial number);
            used for multi-instrument files */
00e   unsigned short checksum; /* 16 bit checksum of structure + data */
       }  KFF_TAG;   // 010h bytes
```

Definition - FRAME HEADER structure

```
/*--------------------------------------------*/
/* Frame Header – Preceeds each Data Frame */
/*--------------------------------------------*/

#define FRAME_HEADER_SIZE 32 /* bytes */
#define MW_FRAME_HEADER_TYPE 4  /* 4 for 17..24 ch,  */
#define K2_FRAME_HEADER_TYPE 3  /* 3 for 1..16 ch */
#define TIMECODE_BYTES 13

/*  FRAME_HEADER   */
Hex
Add
       typedef struct FRAME_HEADER { /* 32 bytes */
000    unsigned char frameType; /* like a version #,
                    /*  3 = 12 channel  */
                    /*  4 = 18 channel  */
                    /*  0 = 12 channel QDR, when instrument code = 0x00 */
001    unsigned char instrumentCode; /* instrument code */
                    /*  0, 0x00 = QDR, when frameType also = 0x00 */
                    /*  9, 0x09 = K2, Mt. Whitney, Old Etna */
                    /* 10, 0x0A = Makalu */
                    /* 20, 0x14 = New Etna */
                    /* 30, 0x1E = Rock */
                    /* 40, 0x28 = SSA2EVT */

002    unsigned short recorderID;
004    unsigned short frameSize; /* # of frame bytes, includes 32
                                            byte header */
006    unsigned long blockTime; /* block time */
00a    unsigned short channelBitMap; /* 1 bit for each ch. in use.
                                            Ch. 1 is bit 0 */
00c    unsigned short streamPar;
                    /* Bits 0-11 = Stream sampling rate, 1..4095 */
                    /* Bits 12-15 = Stream number 0..15 */
00e    unsigned char frameStatus;
                    /*  Bits 0-3 = frame sequence number 0..9  */
                    /* Bit 4    = stream triggered flag. Set if frame exceeds stream
                         trigger level. */
                    /* Bit 5    = Compressed flag. Set if compressed frame. */
                    /* Bits 6-7 = Sample size. 1=16 bits (2 bytes),
                                               2=24 bits (3 bytes)
                                               3=32 bits (4 bytes) */
             /* Samples are expressed as a signed integer of digital counts */
00f    unsigned char frameStatus2;
                    /* Bit 0 = set if issued the ADD SCAN command */
                    /* Bits 1-7 = Makalu:  */
                    /*   1-3 Clip bit – 1st board  */
                    /*   4-6 Clip bit – 2nd board  */
                    /*   7  - Any ADC error  else unused  */
Hex
Add
010    unsigned short msec; /* 0..999 */
012           unsigned char channelBitMap1; /* extended chan bit map, ch17-24 */
013-02f       unsigned char timeCode[TIMECODE_BYTES];
                    /* Time code sampled every millisecond.  */
                    /* Bits 4-7 of timeCode[0] = time code type (TBD) */
                    /*    Bit 3 of timeCode[0] = time code bit sampled at first msec
                                                 of frame,  0=low, 1=high) */
                    /*    Bit 2 of timeCode[0] = time code bit at 2nd msec. */
                    /*   Bit 0 of timeCode[12] = last (100th msec) bit of time code */
       }  //  end  FRAME_HEADER;            // 020 h bytes

/*----------------------------------------------------*/
/*----------------------------------------------------*/
```

Definition – KW HEADER Structure

```
/* kwhead.map */
/* ver 1.10  12 channel K2 Header */
/* ver 1.20  18 channel Mt. Whitney Header */
/* ver 1.30  12 channel K2 Header */
/* ver 1.40  12 channel K2 Header,
   with seismological & Serial Data Stream Parameters,
   Altus 12 channel, QDR, Rock 12 channel  */
/* ver 1.50  18 channel Mt. Whitney Header
   with seismological and Serial Data Stream Parameters
   Altus 18 channel, Rock 18 channel */

/**************************************************/

/* Type Definitions:
   char:  1 byte integer
   short: 2 byte integer
   float: 4 byte floating point (IEEE format)
   long:  4 byte integer

   All structures are 16-bit aligned
*/

// for Etna, K2, Makalu, QDR and Rock
#define MAX_CHANNELS 12

// for Mt. Whitney and Rock
#define MAX_CHANNELS 18
#define FILE_DESCRIPTOR_SIZE 8 /* size of file descriptor, 1st 8 bytes */

struct KW_HEADER {
Hex
Add     offset
       struct RO_PARMS {     /* first 8 bytes is a common file descriptor */
000      0-2   char id[3]; /* = 'KMI' = a Kinematrics file, 0x4B, 0x4D, 0x49 */
003      3     unsigned char instrumentCode;
              /*  9, 0x09 = K2 */
              /* 10, 0x10 = Makalu */
              /* 20, 0x14 = New Etna */
              /* 30, 0x1E = Rock */
              /* 40, 0x28 = SSA2EVT */
004      4-5   unsigned short headerVersion; /* header version * 100 */
              /* 100, 0x64 = 12 channel, old */
              /* 110, 0x6E = 12 channel, old */
              /* 120, 0x78 = 18 channel, old */
              /* 130, 0x82 = 12 channel, old */
              /* 140, 0x8C = 12 channel, K2, Etna, QDR, Rock */
              /* 150, 0x96 = 18 channel, Mt Whitney, Rock */
006      6-7   unsigned short headerBytes; /* size of header following
                                              (includes RW_PARMS) */
Hex
Add  offset
             struct MISC_RO_PARMS {
008      8     unsigned char a2dBits; /* A/D bits per sample; = A2DBITS */
009      9     unsigned char sampleBytes; /* bytes per sample; = 3 */
00a      a     unsigned char restartSource; /* code of restart source:
                    //  0 = unknown
                    //  1 = power switch;
                    //  2 = user command,
                    //  3 = software watchdog,
Hex
```

```
Add  offset
                    //  4 = DSP failure,
                    //  5 = battery failure,
                    //  6 = memory error */
00b      b-d   char bytepad[3]; /* for expansion */

00e      e-f   unsigned short installedChan; /* number of channels in system
                 = (# dsp brds) * (chan per brd) */
010      10-11 unsigned short maxChannels; /* physical number of channels */
012      12-13 unsigned short sysBlkVersion; /* sys block version * 100 */
014      14-15 unsigned short bootBlkVersion; /* boot block version * 100 */
016      16-17 unsigned short appBlkVersion;
                 /* application block version * 100 */
018      18-19 unsigned short dspBlkVersion; /* DSP version * 100 */

               /* System Status */
01a      1a-1b short batteryVoltage; /* voltage * 10; negative value
                                          indicates charging */

01c      1c-1d unsigned short crc; /* 16-bit CRC of entire file,
                                       with this word set to 0xffff */
                     /* NOTE: this parameter is not used at the moment.
                        For integrity checking, the header and each frame
                        are preceded by a structure tag which contains a
                        checksum */
01e      1e-1f unsigned short flags; /* bit 0 = 1 if DSP system error */
20       20-21 short temperature; /* degrees C x 10 */
22       22-27 short wordpad[3]; /* for expansion */

028      28-37 long dwordpad[4]; /* for expansion */
             }; //  end MISC_RO_PARMS

Hex
Add  offset
             struct TIMING_RO_PARMS {
038      38    unsigned char clockSource;
                /* 0 = RTC from cold start
                   1 = keyboard
                   2 = Sync w/ ext. ref. pulse
                   3 = Internal GPS */

039      39    unsigned char gpsStatus;
               /* Bit 0=1 if currently checking for
                          presence of GPS board
                  Bit 1=1 if GPS board present
                  Bit 2=1 if error communicating
                          with GPS
                  Bit 3=1 if failed to lock
                          within an allotted time
                          (gpsMaxTurnOnTime)
                  Bit 4=1 if not locked
                  Bit 5=1 when GPS power is ON
                  Bits 6,7=undefined  */

03a      3a    unsigned char gpsSOH; /* Current state of health;
                 same as Acutime SOH code */
03b      3b-3f unsigned char bytepad[5]; /* for expansion */

040      40-41 unsigned short gpsLockFailCount;
                 /* # of times GPS failed to locked
                 within gpsMaxTurnOnTime */
042      42-43 unsigned short gpsUpdateRTCCount; /* # of times GPS actually
Hex
```

```
Add  offset
                 updated the RTC */
044      44-45 short acqDelay; /* time in msec between actual
                 A/D conversion and DSP output */
046      46-47 short gpsLatitude; /* latitude x 100 , degrees North */
048      48-49 short gpsLongitude; /* longitude x 100, degrees East */
04a      4a-4b short gpsAltitude; /* altitude in meters */
04c      4c-4d unsigned short dacCount; /* TCXO DAC counts */
04e      4e-4f short wordpad; /* for expansion */
050      50-53 short gpsLastDrift[2]; /* in msec.; e.g. 5 = RTC was
                 5 msec faster than GPS */

054      54-5b unsigned long gpsLastTurnOnTime[2];
                 /* time when GPS was last turned on */
05c      5c-63 unsigned long gpsLastUpdateTime[2]; /* time of last RTC update */
064      64-6b unsigned long gpsLastLockTime[2]; /* time of last GPS lock */
06c      6c-7b long dwordpad[4]; /* for expansion */
             }; //  end TIMING_RO_PARMS

//------------------
/*  0x7c  (124)   Bytes used  hex (decimal) */
//------------------
```

BEYOND THIS POINT 12-CHANNEL AND 18-CHANNEL HEADERS ARE DIFFERENT !!!!!!!!!

The **CHANNEL_RO_PARMS** is a dynamic structure. That is, each channel is NOT assigned a specific location, but the recorded channels are entered consecutively.

```
Hex Add
12x 18x offset
             struct CHANNEL_RO_PARMS {
                // 1st recorded channel
07c 07c 0-3     long maxPeak;/* raw sample counts */
080 080 4-7     unsigned long maxPeakOffset; /* offset from start of file */
084 084 8-b     long minPeak;
088 088 c-f     unsigned long minPeakOffset; /* offset from start of file */
08c 08c 10-13   long mean; /* raw sample counts */
090 090 14-17   long aqOffset;
094 094 18-23   long dwordpad[3];/* for expansion */

                // 2nd recorded channel
0a0 0a0 0-3     long maxPeak;/* raw sample counts */
0a4 0a4 4-7     unsigned long maxPeakOffset; /* offset from start of file */
0a8 0a8 8-b     long minPeak;
0ac 0ac c-f     unsigned long minPeakOffset; /* offset from start of file */
0b0 0b0 10-13   long mean; /* raw sample counts */
0b4 0b4 14-17   long aqOffset;
0b8 0b8 18-23   long dwordpad[3];/* for expansion */

                // 3rd recorded channel
0c4 0c4 0-3     long maxPeak;/* raw sample counts */
0c8 0c8 4-7     unsigned long maxPeakOffset; /* offset from start of file */
0cc 0cc 8-b     long minPeak;
0d0 0d0 c-f     unsigned long minPeakOffset; /* offset from start of file */
0d4 0d4 10-13   long mean; /* raw sample counts */
0d8 0d8 14-17   long aqOffset;
0dc 0dc 18-23   long dwordpad[3];/* for expansion */

                // 4th recorded channel
0e8 0e8 0-3     long maxPeak;/* raw sample counts */
0ec 0ec 4-7     unsigned long maxPeakOffset; /* offset from start of file */
0f0 0f0 8-b     long minPeak;
Hex Add
```

```
12x 18x offset
0f4 0f4 c-f     unsigned long minPeakOffset; /* offset from start of file */
0f8 0f8 10-13   long mean; /* raw sample counts */
0fc 0fc 14-17   long aqOffset;
100 100 18-23   long dwordpad[3];/* for expansion */

                // 5th recorded channel
10c 10c 0-3     long maxPeak;/* raw sample counts */
110 110 4-7     unsigned long maxPeakOffset; /* offset from start of file */
114 114 8-b     long minPeak;
118 118 c-f     unsigned long minPeakOffset; /* offset from start of file */
11c 11c 10-13   long mean; /* raw sample counts */
120 120 14-17   long aqOffset;
124 124 18-23   long dwordpad[3];/* for expansion */

                // 6th recorded channel
130 130 0-3     long maxPeak;/* raw sample counts */
134 134 4-7     unsigned long maxPeakOffset; /* offset from start of file */
138 138 8-b     long minPeak;
13c 13c c-f     unsigned long minPeakOffset; /* offset from start of file */
140 140 10-13   long mean; /* raw sample counts */
144 144 14-17   long aqOffset;
148 148 18-23   long dwordpad[3];/* for expansion */

                // 7th recorded channel
154 154 0-3     long maxPeak;/* raw sample counts */
158 158 4-7     unsigned long maxPeakOffset; /* offset from start of file */
15c 15c 8-b     long minPeak;
160 160 c-f     unsigned long minPeakOffset; /* offset from start of file */
164 164 10-13   long mean; /* raw sample counts */
168 168 14-17   long aqOffset;
16c 16c 18-23   long dwordpad[3];/* for expansion */

                // 8th recorded channel
178 178 0-3     long maxPeak;/* raw sample counts */
17c 17c 4-7     unsigned long maxPeakOffset; /* offset from start of file */
180 180 8-b     long minPeak;
184 184 c-f     unsigned long minPeakOffset; /* offset from start of file */
188 188 10-13   long mean; /* raw sample counts */
18c 18c 14-17   long aqOffset;
190 190 18-23   long dwordpad[3];/* for expansion */

                // 9th recorded channel
19c 19c 0-3     long maxPeak;/* raw sample counts */
1a0 1a0 4-7     unsigned long maxPeakOffset; /* offset from start of file */
1a4 1a4 8-b     long minPeak;
1a8 1a8 c-f     unsigned long minPeakOffset; /* offset from start of file */
1ac 1ac 10-13   long mean; /* raw sample counts */
1b0 1b0 14-17   long aqOffset;
1b4 1b4 18-23   long dwordpad[3];/* for expansion */

                // 10th recorded channel
1c0 1c0 0-3     long maxPeak;/* raw sample counts */
1c4 1c4 4-7     unsigned long maxPeakOffset; /* offset from start of file */
1c8 1c8 8-b     long minPeak;
1cc 1cc c-f     unsigned long minPeakOffset; /* offset from start of file */
1d0 1d0 10-13   long mean; /* raw sample counts */
1d4 1d4 14-17   long aqOffset;
1d8 1d8 18-23   long dwordpad[3];/* for expansion */

                // 11th recorded channel
1e4 1e4 0-3     long maxPeak;/* raw sample counts */
Hex Add
```

```
12x 18x offset
1e8 1e8 4-7     unsigned long maxPeakOffset; /* offset from start of file */
1ec 1ec 8-b     long minPeak;
1f0 1f0 c-f     unsigned long minPeakOffset; /* offset from start of file */
1f4 1f4 10-13   long mean; /* raw sample counts */
1f8 1f8 14-17   long aqOffset;
1fc 1fc 18-23   long dwordpad[3];/* for expansion */

                // 12th recorded channel
208 208 0-3     long maxPeak;/* raw sample counts */
20c 20c 4-7     unsigned long maxPeakOffset; /* offset from start of file */
210 210 8-b     long minPeak;
214 214 c-f     unsigned long minPeakOffset; /* offset from start of file */
218 219 10-13   long mean; /* raw sample counts */
21c 21c 14-17   long aqOffset;
220 220 18-23   long dwordpad[3];/* for expansion */

                // 13th recorded channel
--- 22c 0-3     long maxPeak;/* raw sample counts */
--- 230 4-7     unsigned long maxPeakOffset; /* offset from start of file */
--- 234 8-b     long minPeak;
--- 238 c-f     unsigned long minPeakOffset; /* offset from start of file */
--- 23c 10-13   long mean; /* raw sample counts */
--- 240 14-17   long aqOffset;
--- 244 18-23   long dwordpad[3];/* for expansion */

                // 14th recorded channel
--- 250 0-3     long maxPeak;/* raw sample counts */
--- 254 4-7     unsigned long maxPeakOffset; /* offset from start of file */
--- 258 8-b     long minPeak;
--- 25c c-f     unsigned long minPeakOffset; /* offset from start of file */
--- 260 10-13   long mean; /* raw sample counts */
--- 264 14-17   long aqOffset;
--- 268 18-23   long dwordpad[3];/* for expansion */

                // 15th recorded channel
--- 274 0-3     long maxPeak;/* raw sample counts */
--- 278 4-7     unsigned long maxPeakOffset; /* offset from start of file */
--- 27c 8-b     long minPeak;
--- 280 c-f     unsigned long minPeakOffset; /* offset from start of file */
--- 284 10-13   long mean; /* raw sample counts */
--- 288 14-17   long aqOffset;
--- 28c 18-23   long dwordpad[3];/* for expansion */

                // 16th recorded channel
--- 298 0-3     long maxPeak;/* raw sample counts */
--- 29c 4-7     unsigned long maxPeakOffset; /* offset from start of file */
--- 2a0 8-b     long minPeak;
--- 2a4 c-f     unsigned long minPeakOffset; /* offset from start of file */
--- 2a8 10-13   long mean; /* raw sample counts */
--- 2ac 14-17   long aqOffset;
--- 2b0 18-23   long dwordpad[3];/* for expansion */

                // 17th recorded channel
--- 2bc 0-3     long maxPeak;/* raw sample counts */
--- 2c0 4-7     unsigned long maxPeakOffset; /* offset from start of file */
--- 2c4 8-b     long minPeak;
--- 2c8 c-f     unsigned long minPeakOffset; /* offset from start of file */
--- 2cc 10-13   long mean; /* raw sample counts */
--- 2d0 14-17   long aqOffset;
--- 2d4 18-23   long dwordpad[3];/* for expansion */

Hex Add
```

```
12x 18x offset
                // 18th recorded channel
--- 2e0 0-3     long maxPeak;/* raw sample counts */
--- 2e4 4-7     unsigned long maxPeakOffset; /* offset from start of file */
--- 2e8 8-b     long minPeak;
--- 2ec c-f     unsigned long minPeakOffset; /* offset from start of file */
--- 2f0 10-13   long mean; /* raw sample counts */
--- 2f4 14-17   long aqOffset;
--- 2f8 18-23   long dwordpad[3];/* for expansion */
             };   //  end CHANNEL_RO_PARMS


//------------------
/* 12 channels @ 24h(36) = 1b0h(432) 18 channels @ 24h(36) = 288h(648) */
/* 12ch  7ch+1b0h=22ch(556)          18ch  7ch+288h=304h(772) */
//------------------

Hex Add
12x 18x offset
             struct STREAM_RO_PARMS {
22c 304 0-3        unsigned long startTime; /* first sample time,
                     includes PEM */
230 308 4-7        unsigned long triggerTime;
234 30c 8-b        unsigned long duration; /* in # of frames; note:
                     frames may have different sizes */
238 310 c-d        unsigned integer errors;
23a 312 e-f        unsigned integer flags;
                     /* Bit 0  = 1 if functional test,
                                   FT */
                     /*     1 = 1 if sensor response
                                   test, SRT, 080129 */
                     /*     2  = 1 if recorded trigger
                                  data, TRIG DATA */
23c 314 10-11      unsigned integer startTimeMsec;
23e 316 12-13      unsigned integer triggerTimeMsec;
240 318 14-17      unsigned long nscans; /* # of scans in the event */
244 31C 18-1b      Unsigned long triggerbitmap   /* Bit 0 = Channel 1, 7/96 */
248 320 1c-1f      unsigned long pad[1]; /* for expansion */
             };  //  end  STREAM_RO_PARMS
      };  //  end RO_PARMS

//------------------
/* 12x  22ch+20h=24ch(588)   18x  304h+20h=324h(804) */
//------------------


      struct RW_PARMS {

Hex Add
12x 18x offset
             struct MISC_RW_PARMS misc {
24c 324 0-1        unsigned short serialNumber;
24e 326 2-3        unsigned short nchannels; /* number of channels used */

                   #define STN_ID_LENGTH 5
250 328 4-8        char stnID[STN_ID_LENGTH]; /* null terminated */
                   #define COMMENT_LENGTH 33
255 32d 9-29       char comment[COMMENT_LENGTH]; /* NULL terminated */
276 34e 2a-2b      short elevation; /* meters above sea level */
278 350 2c-2f      float latitude; /*  degrees North */
27c 354 30-33      float longitude; /* degrees East */
Hex Add
```

```
12x 18x offset
280 358 34-35      short userCodes[4];

                   /* CRLF output */
288 360 36         unsigned char CRLFCode;
                     /*  0: CRLF Off
                         1: 4800 baud
                         2: 9600 baud
                         3: 19200 baud
                         4: 38400 baud
                         5: 57600 baud */

289 361 37         unsigned char minBatteryVoltage; /* minimum alarm battery
                     voltage x 10 Not Used*/

28a 362 38         unsigned char CRLF_decimation;
                      // not used
                      /* CRLF decimation factor */
                      /* valid 1:1(raw) 1:2 1:4 1:5 1:10 1:20 */
                      /* restrictions on 1:4 @ 250 sps? */
28b 363 39         unsigned char CRLF_irig_type;
                      /* 0:B   1:E (default)   2:H , MCU generated IRIG type,
                                                    Mt. Whitney ONLY*/
28c 364 40-43      unsigned long CRLF_bitmap; /* CRLF bit map */
290 368 44-47      unsigned long channel_bitmap; /* channels selected for
                       acq storage */
294 36c 48         unsigned char CRLF_protocol; /* CRLF, KMI-Agbabian */
295 36d 49-59      char siteID[17]; /* added 7/96 */

             /* Network */
2a6 37e 5a         unsigned char externalTrigger; /* either 1 (on) or 0
                      (off) Not Used*/
2a7 37f 5b         unsigned char networkFlag; /* BIT0 = 0 (Master) or = 1
                       (Slave) Not Used*/
             }; //  end  MISC_RW_PARMS


//------------------
/*  12x  248c+5ch=2a8h(680)     18x   324h+5ch=380h(896) */
//------------------

Hex Add
12x 18x offset
             struct TIMING_RW_PARMS {
2a8 380 0          unsigned char gpsTurnOnInterval; /* minutes between GPS
                       update checking */
2a9 381 1          unsigned char gpsMaxTurnOnTime; /* max time in minutes GPS
                       tries to lock before giving up */
2aa 382 2-7        unsigned char bytepad[6];
2b0 388 8-9        short localOffset; /* time ahead of UTC;
                       if this parameter is non-zero, then all times are in
                       UTC time. If this parameter is zero, then all times are
                       based on local time.
                       0-23 = hours
                       >= 24 = minutes */
2a2 38a a-f        short wordpad[3];
2b6 390 10-1f      long dwordpad[4];
             };  //  end  TIMING_RW_PARMS

//------------------
//  12chan   2a8h+20h=2c8h(712)     18chan   380h+20h=3a0h(928)
//-----------------
```

The **CHANNEL_RW_PARMS** are channel specific, that is a channel's read/write parameters are always in the same location.
#define CHANNEL_ID_LENGTH 5

```
Hex Add
12x 18x offset
             struct CHANNEL_RW_PARMS [MAX_CHANNELS] {
                   // channel 1
2c8 3a0 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
2cd 3a5 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
2ce 3a6 6-7        unsigned short sensorSerialNumberExt; /* high word of s/n */
2d0 3a8 8-9        short north; /* displacement */
2c2 3aa a-b        short east; /* displacement */
2d4 3ac c-d        short up; /* displacement */
2d6 3ae e-f        short altitude;
2d8 3b0 10-11      short azimuth;
2da 3b2 12-13      unsigned short sensorType;
2dc 3b4 14-15      unsigned short sensorSerialNumber; /* low word of s/n */
2de 3b6 16-17      unsigned short gain;

2e0 3b8 18         unsigned char triggertype; /* type of trigger calculation */
                    /* 0 = threshold, default */
                    /* 1 = sta/lta */
2e1 3b9 19         unsigned char iirtrigfilter;
                     /* type of bandpass filter for */
                     /* for trigger, default CSM */
                     /* 0 = iira   IIR bandpass, 1.2 to 20Hz @ 200sps */
                     /* 1 = CSM, classic strong motion, 0.1 to 12.5 Hz @ 200sps */
                     /* 2 = iirc   IIR bandpass, 2.0 to  40Hz at 200sps */
2e2 3ba 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /*  17 valid sta values, code used internally in K2 */
                     /*  0h = 0.1  store as    1 */
                     /*  1  = 0.2              2 */
                     /*  2  = 0.3              3 */
                     /*  3  = 0.4              4 */
                     /*  4  = 0.5              5 */
                     /*  5  = 0.6              6 */
                     /*  6  = 0.8              8 */
                     /*  7  = 1.0             10, default */
                     /*  8  = 1.2             12 */
                     /*  9  = 1.4             14 */
                     /*  a  = 1.6             16 */
                     /*  b  = 1.8             18 */
                     /*  c  = 2.0             20 */
                     /*  d  = 2.5             25 */
                     /*  e  = 3.0             30 */
                     /*  f  = 5.0             50 */
                     /* 10  = 10.0           100 */
2e3 3bb 1b         unsigned char ltaseconds; /* lta seconds */
                     /*  8 valid lta values, code used internally in K2 */
                     /*  0h =  20 store as   20 */
                     /*  1  =  30            30 */
                     /*  2  =  40            40 */
                     /*  3  =  50            50 */
                     /*  4  =  60            60, default */
                     /*  5  =  80            80 */
                     /*  6  = 100           100 */
                     /*  7  = 120           120 */

2e4 3bc 1c-1d      unsigned short sta/ltaratio;
                     /* sta/lta trigger ratio times 10 */
                     /*  12 valid ratio values, code used internally in K2 */
```

```
Hex Add
12x 18x offset
                    /*  0h =   1.5 store as  15 */
                    /*  1  =   2             20 */
                    /*  2  =   3             30 */
                    /*  3  =   4             40, default */
                    /*  4  =   6             60 */
                    /*  5  =   8             80 */
                    /*  6  =  10            100 */
                    /*  7  =  15            150 */
                    /*  8  =  20            200 */
                    /*  9  =  30            300 */
                    /*  a  =  40            400 */
                    /*  b  =  60            500 */
                    /*  c  = 100           1000 */
2e6 3be 1e        unsigned char sta/ltaprecent; /* sta/lta detrigger */
                    /* percent of trigger ratio*/
                    /*  if detrigger percent X trigger ratio is
                        less than 1.2, detrigger ratio will be 1.2 */
                    /*  6 valid detrigger percent values, code used internally */
                    /*  0h =   10% store as  10 */
                    /*  1  =   15            15 */
                    /*  2  =   20            20 */
                    /*  3  =   40            40 */
                    /*  4  =   60            60 */
                    /*  5  =  100           100 */
2e7 3bf 1f        char bytepad1[1]; /* reserved */

2e8 3c0 20-23     float fullscale; /* volts */
2ec 3c4 24-27     float sensitivity; /* in volts per unit (e.g., g's) */
2f0 3c8 28-2b     float damping; /* fraction of critical */
2f4 3cc 2c-2f     float naturalFrequency; /* hz */
2f8 3d0 30-33     float triggerThreshold; /* % of fullscale */
2fc 3d4 34-37     float detriggerThreshold; /* % of fullscale */
300 3d8 38-3b     float alarmTriggerThreshold; /* % of fullscale */
304 3dc 3c-3f     float calCoil   /* g/Volt - EpiSensor */
308 3e0 40        unsigned char range  /* sensor code form EEPROM */
                   /* 1 = 4g */
                   /* 2 = 2g */
                   /* 3 = 1g */
                   /* 4 = 1/2g */
                   /* 5 = 1/4g, etc */
309 3e1 41        unsigned char sensorgain   /* same, but as determined by MCU */
30a 3e2 42-4b     Char bytepad2[10]; /* reserved */

                  // channel 2
314 3ec 0-4       char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
319 3f1 5         char channel; /* physical mapped channel */
                    /* bit 7 = 1, signal inverted */
31a 3f2 6-7       unsigned short sensorSerialNumberExt;
                    /* high word of serial number */
31c 3f4 8-9       short north; /* displacement */
31e 3f6 a-b       short east; /* displacement */
320 3f8 c-d       short up; /* displacement */
322 3fa e-f       short altitude;
324 3fc 10-11     short azimuth;
326 3fe 12-13     unsigned short sensorType;
328 400 14-15     unsigned short sensorSerialNumber;
                    /* low word of serial number */
32a 402 16-17     unsigned short gain;

32c 404 18        unsigned char triggertype; /* type of trigger calculation */
                    /* See channel 1 for codes */
```

```
Hex Add
12x 18x offset
32d 405 19          unsigned char iirtrigfilter; /* type of bandpass filter
                      for trigger, default CSM */
32e 406 1a          unsigned char stasecondsTten; /* sta seconds times 10 */
                      /* See channel 1 for codes */
32f 407 1b          unsigned char ltaseconds; /* lta seconds */
                      /* See channel 1 for codes */
330 408 1c-1d       unsigned short sta/ltaratio; /* sta/lta trigger ratio
                      times 10 */
                      /* See channel 1 for codes */
332 40a 1e          unsigned char sta/ltaprecent; /* sta/lta detrigger */
                      /* percent of trigger ratio*/
                      /* See channel 1 for codes */
333 40b 1f          char bytepad1[1]; /* reserved */

334 40c 20-23       float fullscale; /* volts */
338 410 24-27       float sensitivity; /* in volts per unit (e.g., g's) */
33c 414 28-2b       float damping; /* fraction of critical */
340 418 2c-2f       float naturalFrequency; /* hz */
344 41c 30-33       float triggerThreshold; /* % of fullscale */
348 420 34-37       float detriggerThreshold; /* % of fullscale */
34c 424 38-3b       float alarmTriggerThreshold; /* % of fullscale */
350 428 3c-3f       float calCoil   /* g/Volt - EpiSensor */
354 42c 40          unsigned char range  /* sensor code form EEPROM */
                     /* 1 = 4g */
                     /* 2 = 2g */
                     /* 3 = 1g */
                     /* 4 = 1/2g */
                     /* 5 = 1/4g, etc */
355 42d 41          unsigned char sensorgain   /* same, but as determined by MCU */
356 42e 42-4b       Char bytepad2[10]; /* reserved */

                    // channel 3
360 438 0-4         char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
365 43d 5           char channel; /* physical mapped channel */
                      /* bit 7 = 1, signal inverted */
366 43e 6-7         unsigned short sensorSerialNumberExt;
                      /* high word of serial number */
364 440 8-9         short north; /* displacement */
36a 442 a-b         short east; /* displacement */
36c 444 c-d         short up; /* displacement */
36e 446 e-f         short altitude;
370 448 10-11       short azimuth;
372 44a 12-13       unsigned short sensorType;
374 44c 14-15       unsigned short sensorSerialNumber;
                      /* low word of serial number */
376 44e 16-17       unsigned short gain;

378 450 18          unsigned char triggertype; /* type of trigger calculation */
                      /* See channel 1 for codes */
379 451 19          unsigned char iirtrigfilter; /* type of bandpass filter
                      for trigger, default CSM, See channel 1 for codes */
37a 452 1a          unsigned char stasecondsTten; /* sta seconds times 10 */
                      /* See channel 1 for codes */
37b 453 1b          unsigned char ltaseconds; /* lta seconds */
                      /* See channel 1 for codes */
37c 454 1c-1d       unsigned short sta/ltaratio; /* sta/lta trigger ratio
                      times 10 */
                      /* See channel 1 for codes */
37e 456 1e          unsigned char sta/ltaprecent; /* sta/lta detrigger */
                      /* percent of trigger ratio*/
                      /* See channel 1 for codes */
```

```
Hex Add
12x 18x offset
37f 457 1f         char bytepad1[1]; /* reserved */

380 458 20-23      float fullscale; /* volts */
384 45c 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
388 460 28-2b      float damping; /* fraction of critical */
38c 464 2c-2f      float naturalFrequency; /* hz */
390 468 30-33      float triggerThreshold; /* % of fullscale */
394 46c 34-37      float detriggerThreshold; /* % of fullscale */
398 470 38-3b      float alarmTriggerThreshold; /* % of fullscale */
39c 474 3c-3f      float calCoil   /* g/Volt - EpiSensor */
3a0 478 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
3a1 479 41         unsigned char sensorgain   /* same, but as determined by MCU */
3a2 47a 42-4b      Char bytepad2[10]; /* reserved */

                   // channel 4
3ac 484 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
3a1 489 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
3a2 48a 6-7        unsigned short sensorSerialNumberExt;
                     /* high word of serial number */
3b4 48c 8-9        short north; /* displacement */
3b6 48e a-b        short east; /* displacement */
3b8 490 c-d        short up; /* displacement */
3ba 492 e-f        short altitude;
3bc 494 10-11      short azimuth;
3be 496 12-13      unsigned short sensorType;
3c0 498 14-15      unsigned short sensorSerialNumber;
                     /* low word of serial number */
3c2 49a 16-17      unsigned short gain;

3c4 49c 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
3c5 49d 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
3c6 49e 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
3c7 49f 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
3c8 4a0 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
3ca 4a2 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
3cb 4a3 1f         char bytepad1[1]; /* reserved */

3cc 4a4 20-23      float fullscale; /* volts */
3d0 4a8 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
3d4 4ac 28-2b      float damping; /* fraction of critical */
3d8 4b0 2c-2f      float naturalFrequency; /* hz */
3dc 4b4 30-33      float triggerThreshold; /* % of fullscale */
3e0 4b8 34-37      float detriggerThreshold; /* % of fullscale */
3e4 4bc 38-3b      float alarmTriggerThreshold; /* % of fullscale */
3e8 4c0 3c-3f      float calCoil   /* g/Volt - EpiSensor */
3ec 4c4 40         unsigned char range  /* sensor code form EEPROM */
```

```
Hex Add
12x 18x offset
                          /* 1 = 4g */
                          /* 2 = 2g */
                          /* 3 = 1g */
                          /* 4 = 1/2g */
                          /* 5 = 1/4g, etc */
3ed 4c5 41               unsigned char sensorgain   /* same, but as determined by MCU */
3ee 4c6 42-4b            Char bytepad2[10]; /* reserved */

                         // channel 5
3f8 4d0 0-4              char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
3fd 4d5 5                char channel; /* physical mapped channel */
                           /* bit 7 = 1, signal inverted */
3fe 4d6 6-7              unsigned short sensorSerialNumberExt;
                           /* high word of serial number */
400 4d8 8-9              short north; /* displacement */
402 4da a-b              short east; /* displacement */
404 4dc c-d              short up; /* displacement */
406 4de e-f              short altitude;
408 4e0 10-11            short azimuth;
40a 4e2 12-13            unsigned short sensorType;
40c 4e4 14-15            unsigned short sensorSerialNumber;
                          /* low word of serial number */
40e 4e6 16-17            unsigned short gain;

410 4e8 18               unsigned char triggertype; /* type of trigger calculation */
                           /* See channel 1 for codes */
411 4e9 19               unsigned char iirtrigfilter; /* type of bandpass filter
                           for trigger, default CSM */
                           /* See channel 1 for codes */
412 4ea 1a               unsigned char stasecondsTten; /* sta seconds times 10 */
                           /* See channel 1 for codes */
413 4eb 1b               unsigned char ltaseconds; /* lta seconds */
                           /* See channel 1 for codes */
414 4ec 1c-1d            unsigned short sta/ltaratio; /* sta/lta trigger ratio
                           times 10 */
                           /* See channel 1 for codes */
416 4ee 1e               unsigned char sta/ltaprecent; /* sta/lta detrigger */
                           /* percent of trigger ratio*/
                           /* See channel 1 for codes */
417 4ef 1f               char bytepad1[1]; /* reserved */

418 4f0 20-23            float fullscale; /* volts */
41c 4f4 24-27            float sensitivity; /* in volts per unit (e.g., g's) */
420 4f8 28-2b            float damping; /* fraction of critical */
424 4fc 2c-2f            float naturalFrequency; /* hz */
428 500 30-33            float triggerThreshold; /* % of fullscale */
42v 504 34-37            float detriggerThreshold; /* % of fullscale */
430 508 38-3b            float alarmTriggerThreshold; /* % of fullscale */
434 50c 3c-3f            float calCoil   /* g/Volt - EpiSensor */
438 510 40               unsigned char range  /* sensor code form EEPROM */
                          /* 1 = 4g */
                          /* 2 = 2g */
                          /* 3 = 1g */
                          /* 4 = 1/2g */
                          /* 5 = 1/4g, etc */
439 511 41               unsigned char sensorgain   /* same, but as determined by MCU */
43a 512 42-4b            Char bytepad2[10]; /* reserved */

                         // channel 6
444 51c 0-4              char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
449 521 5                char channel; /* physical mapped channel */
```

```
Hex Add
12x 18x offset
                         /* bit 7 = 1, signal inverted */
44a 522 6-7           unsigned short sensorSerialNumberExt;
                         /* high word of serial number */
44c 524 8-9           short north; /* displacement */
44e 526 a-b           short east; /* displacement */
450 528 c-d           short up; /* displacement */
452 52a e-f           short altitude;
454 52c 10-11         short azimuth;
456 52e 12-13         unsigned short sensorType;
458 530 14-15         unsigned short sensorSerialNumber;
                         /* low word of serial number */
45a 532 16-17         unsigned short gain;

45c 534 18            unsigned char triggertype; /* type of trigger calculation */
                         /* See channel 1 for codes */
45d 535 19            unsigned char iirtrigfilter; /* type of bandpass filter
                         for trigger, default CSM */
                         /* See channel 1 for codes */
45e 536 1a            unsigned char stasecondsTten; /* sta seconds times 10 */
                         /* See channel 1 for codes */
45f 537 1b            unsigned char ltaseconds; /* lta seconds */
                         /* See channel 1 for codes */
460 538 1c-1d         unsigned short sta/ltaratio; /* sta/lta trigger ratio
                         times 10 */
                         /* See channel 1 for codes */
462 53a 1e            unsigned char sta/ltaprecent; /* sta/lta detrigger */
                         /* percent of trigger ratio*/
                         /* See channel 1 for codes */
463 53b 1f            char bytepad1[1]; /* reserved */

464 53c 20-23         float fullscale; /* volts */
468 540 24-27         float sensitivity; /* in volts per unit (e.g., g's) */
46c 544 28-2b         float damping; /* fraction of critical */
470 548 2c-2f         float naturalFrequency; /* hz */
474 54c 30-33         float triggerThreshold; /* % of fullscale */
478 550 34-37         float detriggerThreshold; /* % of fullscale */
47c 554 38-3b         float alarmTriggerThreshold; /* % of fullscale */
480 558 3c-3f         float calCoil   /* g/Volt - EpiSensor */
484 55c 40            unsigned char range  /* sensor code form EEPROM */
                        /* 1 = 4g */
                        /* 2 = 2g */
                        /* 3 = 1g */
                        /* 4 = 1/2g */
                        /* 5 = 1/4g, etc */
485 55d 41            unsigned char sensorgain   /* same, but as determined by MCU */
486 55e 42-4b         Char bytepad2[10]; /* reserved */

                      // channel 7
490 568 0-4           char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
495 56d 5             char channel; /* physical mapped channel */
                         /* bit 7 = 1, signal inverted */
496 56e 6-7           unsigned short sensorSerialNumberExt;
                         /* high word of serial number */
498 570 8-9           short north; /* displacement */
49a 572 a-b           short east; /* displacement */
49c 574 c-d           short up; /* displacement */
49e 576 e-f           short altitude;
4a0 578 10-11         short azimuth;
4a2 57a 12-13         unsigned short sensorType;
4a4 57c 14-15         unsigned short sensorSerialNumber;
                         /* low word of serial number */
```

```
Hex Add
12x 18x offset
4a6 57e 16-17      unsigned short gain;

4a8 580 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
4a9 581 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
4aa 582 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
4ab 583 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
4ac 584 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
4ae 586 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
4af 587 1f         char bytepad1[1]; /* reserved */

4b0 588 20-23      float fullscale; /* volts */
4b4 58c 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
4b8 590 28-2b      float damping; /* fraction of critical */
4bc 594 2c-2f      float naturalFrequency; /* hz */
4c0 598 30-33      float triggerThreshold; /* % of fullscale */
4c4 59c 34-37      float detriggerThreshold; /* % of fullscale */
4c8 5a0 38-3b      float alarmTriggerThreshold; /* % of fullscale */
4cc 5a4 3c-3f      float calCoil   /* g/Volt - EpiSensor */
4d0 5a8 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
4d1 5a9 41         unsigned char sensorgain   /* same, but as determined by MCU */
4d2 5aa 42-4b      Char bytepad2[10]; /* reserved */

                   // channel 8
4dc 5b4 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
4e1 5b9 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
4e2 5ba 6-7        unsigned short sensorSerialNumberExt;
                     /* high word of serial number */
4e4 5bc 8-9        short north; /* displacement */
4e6 5be a-b        short east; /* displacement */
4e8 5c0 c-d        short up; /* displacement */
4ea 5c2 e-f        short altitude;
4ec 5c4 10-11      short azimuth;
4ee 5c6 12-13      unsigned short sensorType;
4f0 5c8 14-15      unsigned short sensorSerialNumber;
                     /* low word of serial number */
4f2 5ca 16-17      unsigned short gain;
4f4 5cc 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
4f5 5cd 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
4f6 5ce 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
4f7 5cf 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
4f8 5d0 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio x 10 */
```

```
Hex Add
12x 18x offset
                     /* See channel 1 for codes */
4fa 5d2 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
4fb 5d3 1f         char bytepad1[1]; /* reserved */

4fc 5d4 20-23      float fullscale; /* volts */
500 5d8 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
504 5dc 28-2b      float damping; /* fraction of critical */
508 5e0 2c-2f      float naturalFrequency; /* hz */
50c 5e4 30-33      float triggerThreshold; /* % of fullscale */
510 5e8 34-37      float detriggerThreshold; /* % of fullscale */
514 5ec 38-3b      float alarmTriggerThreshold; /* % of fullscale */
518 5f0 3c-3f      float calCoil   /* g/Volt - EpiSensor */
51c 5f4 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
51d 5f5 41         unsigned char sensorgain   /* same, but as determined by MCU */
51e 5f6 42-4b      Char bytepad2[10]; /* reserved */

                   // channel 9
528 600 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
52d 605 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
52e 606 6-7        unsigned short sensorSerialNumberExt;
                    /* high word of serial number */
530 608 8-9        short north; /* displacement */
532 60a a-b        short east; /* displacement */
534 60c c-d        short up; /* displacement */
536 60e e-f        short altitude;
538 610 10-11      short azimuth;
53a 612 12-13      unsigned short sensorType;
53c 614 14-15      unsigned short sensorSerialNumber;
                     /* low word of serial number */
53e 616 16-17      unsigned short gain;

540 618 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
541 619 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
542 61a 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
543 61b 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
544 61c 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio x 10 */
                     /* See channel 1 for codes */
546 61e 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
547 61f 1f         char bytepad1[1]; /* reserved */

548 620 20-23      float fullscale; /* volts */
54c 624 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
550 628 28-2b      float damping; /* fraction of critical */
554 62c 2c-2f      float naturalFrequency; /* hz */
558 630 30-33      float triggerThreshold; /* % of fullscale */
55c 634 34-37      float detriggerThreshold; /* % of fullscale */
```

```
Hex Add
12x 18x offset
560 638 38-3b      float alarmTriggerThreshold; /* % of fullscale */
564 63c 3c-3f      float calCoil   /* g/Volt - EpiSensor */
568 640 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
569 641 41         unsigned char sensorgain   /* same, but as determined by MCU */
56a 632 42-4b      Char bytepad2[10]; /* reserved */

                   // channel 10
574 64c 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
579 651 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
57a 652 6-7        unsigned short sensorSerialNumberExt;
                    /* high word of serial number */
57c 654 8-9        short north; /* displacement */
57e 656 a-b        short east; /* displacement */
580 658 c-d        short up; /* displacement */
582 65a e-f        short altitude;
584 65c 10-11      short azimuth;
586 65e 12-13      unsigned short sensorType;
588 660 14-15      unsigned short sensorSerialNumber;
                    /* low word of serial number */
58a 662 16-17      unsigned short gain;

58c 664 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
58d 665 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
58e 666 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
58f 667 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
590 668 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
592 66a 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
593 66b 1f         char bytepad1[1]; /* reserved */

594 66c 20-23      float fullscale; /* volts */
598 670 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
59c 674 28-2b      float damping; /* fraction of critical */
5a0 678 2c-2f      float naturalFrequency; /* hz */
5a4 67c 30-33      float triggerThreshold; /* % of fullscale */
5a8 680 34-37      float detriggerThreshold; /* % of fullscale */
5ac 684 38-3b      float alarmTriggerThreshold; /* % of fullscale */
5b0 688 3c-3f      float calCoil   /* g/Volt - EpiSensor */
5b4 68c 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
5b5 68d 41         unsigned char sensorgain   /* same, but as determined by MCU */
5b6 68e 42-4b      Char bytepad2[10]; /* reserved */
```

```
                   // channel 11
Hex Add
12x 18x offset
5c0 698 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
5c5 69d 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
5c6 69e 6-7        unsigned short sensorSerialNumberExt;
                    /* high word of serial number */
5c8 6a0 8-9        short north; /* displacement */
5ca 6a2 a-b        short east; /* displacement */
5cc 6a4 c-d        short up; /* displacement */
5ce 6a6 e-f        short altitude;
5d0 6a8 10-11      short azimuth;
5d2 6aa 12-13      unsigned short sensorType;
5d4 6ac 14-15      unsigned short sensorSerialNumber;
                    /* low word of serial number */
5d6 6ae 16-17      unsigned short gain;

5d8 6b0 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
5d9 6b1 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
5da 6b2 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
5db 6b3 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
5dc 6b4 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
5de 6b6 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
5df 6b7 1f         char bytepad1[1]; /* reserved */

5e0 6b8 20-23      float fullscale; /* volts */
5e4 6bc 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
5e8 6c0 28-2b      float damping; /* fraction of critical */
5ec 6c4 2c-2f      float naturalFrequency; /* hz */
5f0 6c8 30-33      float triggerThreshold; /* % of fullscale */
5f4 6cc 34-37      float detriggerThreshold; /* % of fullscale */
5f8 6d0 38-3b      float alarmTriggerThreshold; /* % of fullscale */
5fc 6d4 3c-3f      float calCoil   /* g/Volt - EpiSensor */
600 6d8 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
601 6d9 41         unsigned char sensorgain   /* same, but as determined by MCU */
602 6da 42-4b      Char bytepad2[10]; /* reserved */

                   // channel 12
60c 6e4 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
611 6e9 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
612 6ea 6-7        unsigned short sensorSerialNumberExt;
                    /* high word of serial number */
614 6ec 8-9        short north; /* displacement */
616 6ee a-b        short east; /* displacement */
618 6f0 c-d        short up; /* displacement */
61a 6f2 e-f        short altitude;
61c 6f4 10-11      short azimuth;
```

```
Hex Add
12x 18x offset
61e 6f6 12-13      unsigned short sensorType;
620 6f8 14-15      unsigned short sensorSerialNumber;
                    /* low word of serial number */
622 6fa 16-17      unsigned short gain;

624 6fc 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
625 6fd 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
626 6fe 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
627 6ff 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
628 700 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
62a 702 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
62b 704 1f         char bytepad1[1]; /* reserved */

62c 704 20-23      float fullscale; /* volts */
630 708 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
634 70c 28-2b      float damping; /* fraction of critical */
638 710 2c-2f      float naturalFrequency; /* hz */
63c 714 30-33      float triggerThreshold; /* % of fullscale */
640 718 34-37      float detriggerThreshold; /* % of fullscale */
644 71c 38-3b      float alarmTriggerThreshold; /* % of fullscale */
648 720 3c-3f      float calCoil   /* g/Volt - EpiSensor */
64c 724 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
64d 715 41         unsigned char sensorgain   /* same, but as determined by MCU */
64e 726 42-4b      Char bytepad2[10]; /* reserved */

                   // channel 13
--- 730 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
--- 735 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
--- 736 6-7        unsigned short sensorSerialNumberExt;
                    /* high word of serial number */
--- 738 8-9        short north; /* displacement */
--- 73a a-b        short east; /* displacement */
--- 73c c-d        short up; /* displacement */
--- 73e e-f        short altitude;
--- 740 10-11      short azimuth;
--- 742 12-13      unsigned short sensorType;
--- 744 14-15      unsigned short sensorSerialNumber;
                    /* low word of serial number */
--- 746 16-17      unsigned short gain;

--- 748 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
--- 749 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
--- 74a 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
```

```
Hex Add
12x 18x offset
                     /* See channel 1 for codes */
--- 74b 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
--- 74c 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
--- 74e 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
--- 74f 1f         char bytepad1[1]; /* reserved */

--- 750 20-23      float fullscale; /* volts */
--- 754 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
--- 758 28-2b      float damping; /* fraction of critical */
--- 75c 2c-2f      float naturalFrequency; /* hz */
--- 760 30-33      float triggerThreshold; /* % of fullscale */
--- 764 34-37      float detriggerThreshold; /* % of fullscale */
--- 768 38-3b      float alarmTriggerThreshold; /* % of fullscale */
--- 76c 3c-3f      float calCoil   /* g/Volt - EpiSensor */
--- 770 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
--- 771 41         unsigned char sensorgain   /* same, but as determined by MCU */
--- 772 42-4b      Char bytepad2[10]; /* reserved */

                   // channel 14
--- 77c 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
--- 781 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
--- 782 6-7        unsigned short sensorSerialNumberExt;
                    /* high word of serial number */
--- 784 8-9        short north; /* displacement */
--- 786 a-b        short east; /* displacement */
--- 788 c-d        short up; /* displacement */
--- 78a e-f        short altitude;
--- 78c 10-11      short azimuth;
--- 78e 12-13      unsigned short sensorType;
--- 790 14-15      unsigned short sensorSerialNumber;
                     /* low word of serial number */
--- 792 16-17      unsigned short gain;

--- 794 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
--- 795 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
--- 796 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
--- 797 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
--- 798 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
--- 79a 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
--- 79a 1f         char bytepad1[1]; /* reserved */
```

```
Hex Add
12x 18x offset
--- 79c 20-23      float fullscale; /* volts */
--- 7a0 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
--- 7a4 28-2b      float damping; /* fraction of critical */
--- 7a8 2c-2f      float naturalFrequency; /* hz */
--- 7ac 30-33      float triggerThreshold; /* % of fullscale */
--- 7b0 34-37      float detriggerThreshold; /* % of fullscale */
--- 7b4 38-3b      float alarmTriggerThreshold; /* % of fullscale */
--- 7b8 3c-3f      float calCoil   /* g/Volt - EpiSensor */
--- 7bc 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
--- 7bd 41         unsigned char sensorgain   /* same, but as determined by MCU */
--- 7be 42-4b      Char bytepad2[10]; /* reserved */


                   // channel 15
--- 7c8 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
--- 7cd 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
--- 7ce 6-7        unsigned short sensorSerialNumberExt;
                     /* high word of serial number */
--- 7d0 8-9        short north; /* displacement */
--- 7d2 a-b        short east; /* displacement */
--- 7d4 c-d        short up; /* displacement */
--- 7d6 e-f        short altitude;
--- 7d8 10-11      short azimuth;
--- 7da 12-13      unsigned short sensorType;
--- 7dc 14-15      unsigned short sensorSerialNumber;
                     /* low word of serial number */
--- 7de 16-17      unsigned short gain;

--- 7e0 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
--- 7e1 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
--- 7e2 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
--- 7e3 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
--- 7e4 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
--- 7e6 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
--- 7e7 1f         char bytepad1[1]; /* reserved */
--- 7e8 20-23      float fullscale; /* volts */
--- 7ec 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
--- 7f0 28-2b      float damping; /* fraction of critical */
--- 7f4 2c-2f      float naturalFrequency; /* hz */
--- 7f8 30-33      float triggerThreshold; /* % of fullscale */
--- 7fc 34-37      float detriggerThreshold; /* % of fullscale */
--- 800 38-3b      float alarmTriggerThreshold; /* % of fullscale */
--- 804 3c-3f      float calCoil   /* g/Volt - EpiSensor */
--- 808 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
```

```
Hex Add
12x 18x offset
                      /* 3 = 1g */
                      /* 4 = 1/2g */
                      /* 5 = 1/4g, etc */
--- 809 41          unsigned char sensorgain   /* same, but as determined by MCU */
--- 80a 42-4b       Char bytepad2[10]; /* reserved */

                    // channel 16
--- 814 0-4         char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
--- 819 5           char channel; /* physical mapped channel */
                      /* bit 7 = 1, signal inverted */
--- 81a 6-7         unsigned short sensorSerialNumberExt;
                      /* high word of serial number */
--- 81c 8-9         short north; /* displacement */
--- 81e a-b         short east; /* displacement */
--- 820 c-d         short up; /* displacement */
--- 822 e-f         short altitude;
--- 824 10-11       short azimuth;
--- 826 12-13       unsigned short sensorType;
--- 828 14-15       unsigned short sensorSerialNumber;
                      /* low word of serial number */
--- 82a 16-17       unsigned short gain;

--- 82c 18          unsigned char triggertype; /* type of trigger calculation */
                      /* See channel 1 for codes */
--- 82d 19          unsigned char iirtrigfilter; /* type of bandpass filter
                      for trigger, default CSM */
                      /* See channel 1 for codes */
--- 82e 1a          unsigned char stasecondsTten; /* sta seconds times 10 */
                      /* See channel 1 for codes */
--- 82f 1b          unsigned char ltaseconds; /* lta seconds */
                      /* See channel 1 for codes */
--- 830 1c-1d       unsigned short sta/ltaratio; /* sta/lta trigger ratio
                      times 10 */
                      /* See channel 1 for codes */
--- 832 1e          unsigned char sta/ltaprecent; /* sta/lta detrigger */
                      /* percent of trigger ratio*/
                      /* See channel 1 for codes */
--- 833 1f          char bytepad1[1]; /* reserved */

--- 834 20-23       float fullscale; /* volts */
--- 838 24-27       float sensitivity; /* in volts per unit (e.g., g's) */
--- 83c 28-2b       float damping; /* fraction of critical */
--- 840 2c-2f       float naturalFrequency; /* hz */
--- 844 30-33       float triggerThreshold; /* % of fullscale */
--- 848 34-37       float detriggerThreshold; /* % of fullscale */
--- 84c 38-3b       float alarmTriggerThreshold; /* % of fullscale */
--- 850 3c-3f       float calCoil   /* g/Volt - EpiSensor */
--- 854 40          unsigned char range  /* sensor code form EEPROM */
                      /* 1 = 4g */
                      /* 2 = 2g */
                      /* 3 = 1g */
                      /* 4 = 1/2g */
                      /* 5 = 1/4g, etc */
--- 855 41          unsigned char sensorgain   /* same, but as determined by MCU */
--- 856 42-4b       Char bytepad2[10]; /* reserved */

                    // channel 17
--- 860 0-4         char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
--- 865 5           char channel; /* physical mapped channel */
                      /* bit 7 = 1, signal inverted */
--- 866 6-7         unsigned short sensorSerialNumberExt; /* high word of S/N */
```

```
Hex Add
12x 18x offset
--- 868 8-9        short north; /* displacement */
--- 86a a-b        short east; /* displacement */
--- 86c c-d        short up; /* displacement */
--- 86e e-f        short altitude;
--- 870 10-11      short azimuth;
--- 872 12-13      unsigned short sensorType;
--- 874 14-15      unsigned short
                   sensorSerialNumber;
                     /* low word of serial number */
--- 876 16-17      unsigned short gain;

--- 878 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
--- 879 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
--- 87a 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
--- 87b 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
--- 87c 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
--- 87e 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
--- 87f 1f         char bytepad1[1]; /* reserved */

--- 880 20-23      float fullscale; /* volts */
--- 884 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
--- 888 28-2b      float damping; /* fraction of critical */
--- 88c 2c-2f      float naturalFrequency; /* hz */
--- 890 30-33      float triggerThreshold; /* % of fullscale */
--- 894 34-37      float detriggerThreshold; /* % of fullscale */
--- 898 38-3b      float alarmTriggerThreshold; /* % of fullscale */
--- 89c 3c-3f      float calCoil   /* g/Volt - EpiSensor */
--- 8a0 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
--- 8a1 41         unsigned char sensorgain   /* same, but as determined by MCU */
--- 8a2 42-4b      Char bytepad2[10]; /* reserved */

                   // channel 18
--- 8ac 0-4        char id[CHANNEL_ID_LENGTH]; /* NULL terminated */
--- 8b1 5          char channel; /* physical mapped channel */
                     /* bit 7 = 1, signal inverted */
--- 8b2 6-7        unsigned short sensorSerialNumberExt;
                     /* high word of serial number */
--- 8b4 8-9        short north; /* displacement */
--- 8b6 a-b        short east; /* displacement */
--- 8b8 c-d        short up; /* displacement */
--- 8ba e-f        short altitude;
--- 8bc 10-11      short azimuth;
--- 8be 12-13      unsigned short sensorType;
--- 8c0 14-15      unsigned short sensorSerialNumber;
                     /* low word of serial number */
--- 8c2 16-17      unsigned short gain;
```

```
Hex Add
12x 18x offset
--- 8c4 18         unsigned char triggertype; /* type of trigger calculation */
                     /* See channel 1 for codes */
--- 8c5 19         unsigned char iirtrigfilter; /* type of bandpass filter
                     for trigger, default CSM */
                     /* See channel 1 for codes */
--- 8c6 1a         unsigned char stasecondsTten; /* sta seconds times 10 */
                     /* See channel 1 for codes */
--- 8c7 1b         unsigned char ltaseconds; /* lta seconds */
                     /* See channel 1 for codes */
--- 8c8 1c-1d      unsigned short sta/ltaratio; /* sta/lta trigger ratio
                     times 10 */
                     /* See channel 1 for codes */
--- 8ca 1e         unsigned char sta/ltaprecent; /* sta/lta detrigger */
                     /* percent of trigger ratio*/
                     /* See channel 1 for codes */
--- 8cb 1f         char bytepad1[1]; /* reserved */
--- 8cc 20-23      float fullscale; /* volts */
--- 8d0 24-27      float sensitivity; /* in volts per unit (e.g., g's) */
--- 8d4 28-2b      float damping; /* fraction of critical */
--- 8d8 2c-2f      float naturalFrequency; /* hz */
--- 8dc 30-33      float triggerThreshold; /* % of fullscale */
--- 8e0 34-37      float detriggerThreshold; /* % of fullscale */
--- 8e4 38-3b      float alarmTriggerThreshold; /* % of fullscale */

--- 8e8 3c-3f      float calCoil   /* g/Volt - EpiSensor */
--- 8ec 40         unsigned char range  /* sensor code form EEPROM */
                    /* 1 = 4g */
                    /* 2 = 2g */
                    /* 3 = 1g */
                    /* 4 = 1/2g */
                    /* 5 = 1/4g, etc */
--- 8ed 41         unsigned char sensorgain   /* same, but as determined by MCU */
--- 8ee 42-4b      Char bytepad2[10]; /* reserved */
             }; //  end  CHANNEL_RW_PARMS


//------------------
/*  12chan   2c8h+ (12*76) 390h(912)=658H(1624)
    18chan   3a0h+ (18*76) 558h(1368)=8f8h(2296)  */
//------------------

Hex Add
12x 18x offset
             struct STREAM_RW_PARMS {
658 8f8 0          unsigned char filterFlag; /* BIT0 = 1 if filtered data;
                                                BIT1 = 1 if auto FT after Event
                                                BIT2 = 1 if compressed */
659 8f9 1          unsigned char primaryStorage; /* = 0 for drive A: ,etc. */
65a 8fa 2          unsigned char secondaryStorage; /* = 1 for drive B:, etc. */
65b 8fb 3-7        unsigned char bytepad[5]; /* for expansion */
660 900 8-9        unsigned short eventNumber; /* *NOT USED* */
662 902 a-b        unsigned short sps; /* sampling rate */
664 904 c-d        unsigned short apw;
                     /* array propagation window, in seconds*/
666 906 e-f        unsigned short preEvent; /* in seconds */
668 908 10-11      unsigned short postEvent; /* in seconds */
66a 90a 12-13      unsigned short minRunTime; /* in seconds */
66c 90c 14-15      short VotesToTrigger; /* signed number */
66e 90e 16-17      short VotesToDetrigger;

670 910 18         char bytepada;
```

```
Hex Add
12x 18x offset
671 911 19         unsigned char FilterType; /* FirB: 0 regular, 1 causal */
672 912 1a         unsigned char DataFmt; /* Serial Data Stream */
                    /* 0 = uncompressed
                       1 = compressed  */
673 913 1b         char Reserved;
674 914 1c-1d      short Timeout;
                    /* Serial Data Stream mode:    */
                    /*   1 - 0   default           */
                    /*   2 - 6..32767              */
                    /*          send 'continue cmd'*/
                    /*   3 - -1 cmd for each block */

676 916 1e-1f      unsigned short TxBlkSize; /* Serial Data Strea (unused)  */
678 918 20-21      unsigned short BufferSize; /* Serial Data Stream:  */
                     /* size of #SSTRBUF.BINfile in disk */
                     /* number of 16,384 byte blocks */
                     /* default= 64 - 1Meg */
67a 91a 22-23      unsigned short SampleRate; /* Serial Data Stream sample rate */
67c 91c 24-27      long TxChanMap; /* Serial Data Stream:  */
                     /* select up to 6 channels */
680 980 28-2f      long dwordpad[2]; /* for expansion */
             };  //  end  STREAM_RW_PARMS

//------------------
/*  12chan   658h+ 30H= 688h(1672)     18chan   8f8h + 30h= 928h(2344) */
//------------------

  /* Voter info: one entry for each channel, network, external, keyboard selected */
  /* A dynamic structure, that is each source is NOT assigned a specific location */
  /* and they can be in any order. */
  /* 'unsigned char number' is only valid if the 'unsigned char type' is 'C' */

  #define STREAM_MAX_VOTERS  (MAX_CHANNELS +3)
  /* 12 channel:  12 + 3 = 15, 18 channel:  18 + 3 = 21 */

             struct VOTER_INFO [STREAM_MAX_VOTERS] {

Hex Add
12x 18x offset
                 // voter 1
688 928 0          unsigned char type; /* voter type code */
                                      /* 'C' channel */
                                      /* 'E' external */
                                      /* 'K' keyboard */
                                      /* 'N' network */
689 929 1          unsigned char number; /* channel number, stream number */
68a 92a 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 2
68c 92c 0          unsigned char type; /* voter type code */
68d 92d 1          unsigned char number; /* channel number */
68e 92e 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 3
690 930 0          unsigned char type; /* voter type code */
691 931 1          unsigned char number; /* channel number */
692 932 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 4
694 934 0          unsigned char type; /* voter type code */
695 935 1          unsigned char number; /* channel number */
```

```
Hex Add
12x 18x offset
696 936 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 5
698 938 0          unsigned char type; /* voter type code */
699 939 1          unsigned char number; /* channel number */
69a 93a 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 6
69c 93c 0          unsigned char type; /* voter type code */
69d 93d 1          unsigned char number; /* channel number */
69e 93e 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 7
6a0 940 0          unsigned char type; /* voter type code */
6a1 941 1          unsigned char number; /* channel number */
6a2 942 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 8
6a4 944 0          unsigned char type; /* voter type code */
6a5 945 1          unsigned char number; /* channel number */
6a6 946 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 9
6a8 948 0          unsigned char type; /* voter type code */
6a9 949 1          unsigned char number; /* channel number */
6aa 94a 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 10
6ac 94c 0          unsigned char type; /* voter type code */
6ad 94d 1          unsigned char number; /* channel number */
6ae 94e 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 11
6b0 950 0          unsigned char type; /* voter type code */
6b1 951 1          unsigned char number; /* channel number */
6b2 952 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 12
6b4 954 0          unsigned char type; /* voter type code */
6b5 955 1          unsigned char number; /* channel number */
6b6 956 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 13
6b8 958 0          unsigned char type; /* voter type code */
6b9 959 1          unsigned char number; /* channel number */
6ba 95a 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 14
6bc 95c 0          unsigned char type; /* voter type code */
6bd 95d 1          unsigned char number; /* channel number */
6be 95e 2-3        short weight; /* voting weight: range is -100 to 100 */
                   // voter 15
6c0 960 0          unsigned char type; /* voter type code */
6c1 961 1          unsigned char number; /* channel number */
6c2 962 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 16
--- 964 0          unsigned char type; /* voter type code */
--- 965 1          unsigned char number; /* channel number */
--- 966 2-3        short weight; /* voting weight: range is -100 to 100 */
```

```
                   // voter 17
Hex Add
12x 18x offset
--- 968 0          unsigned char type; /* voter type code */
--- 969 1          unsigned char number; /* channel number */
--- 96a 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 18
--- 96c 0          unsigned char type; /* voter type code */
--- 96d 1          unsigned char number; /* channel number */
--- 96e 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 19
--- 970 0          unsigned char type; /* voter type code */
--- 971 1          unsigned char number; /* channel number */
--- 972 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 20
--- 974 0          unsigned char type; /* voter type code */
--- 975 1          unsigned char number; /* channel number */
--- 976 2-3        short weight; /* voting weight: range is -100 to 100 */

                   // voter 21
--- 978 0          unsigned char type; /* voter type code */
--- 979 1          unsigned char number; /* channel number */
--- 97a 2-3        short weight; /* voting weight: range is -100 to 100 */
             };  //  end  VOTER_INFO


//------------------
/*  12chan   688 + (15*4)[3ch(60)] = 6c4h(1732)
    18chan   928h + (21*4)[54h(84)] = 97ch(2428)  */
//------------------

          #define MODEM_INITCMD_LENGTH 64
          #define MODEM_DIALPREFIX_LENGTH 16
          #define MODEM_DIALSUFFIX_LENGTH 16
          #define MODEM_HANGUPCMD_LENGTH 16
          #define MODEM_AUTOANSWERON_LENGTH 16
          #define MODEM_AUTOANSWEROFF_LENGTH 16
          #define MODEM_PHONES 4
          #define MODEM_PHONENUMBER_LENGTH 24

Hex Add
12x 18x offset
             struct MODEM_RW_PARMS {
6c4 97c 0-3f    char initCmd[MODEM_INITCMD_LENGTH]; /* initialization string */
704 9bc 40-4f   char dialingPrefix [MODEM_DIALPREFIX_LENGTH];
                  /* dialing prefix */
714 9cc 50-5f   char dialingSuffix [MODEM_DIALSUFFIX_LENGTH];
724 9dc 60-6f   char hangupCmd [MODEM_HANGUPCMD_LENGTH];
734 9ec 70-7f   char autoAnswerOnCmd [MODEM_AUTOANSWERON_LENGTH];
                  /* NOT USED */
744 9fc 80-8f   char autoAnswerOffCmd [MODEM_AUTOANSWEROFF_LENGTH];
                  /* NOT USED */
754 a0c 1 90-a7 char phoneNumber[MODEM_PHONES] [MODEM_PHONENUMBER_LENGTH];
                  /* 4 phone numbers, ONLY 2 are currently used */
76c a24 2 a8-bf
784 a3c 3 c0-d7
79c a54 4 d8-ef
7b4 a6c f0      unsigned char waitForConnection; /* secs */
7b5 a6d f1      unsigned char pauseBetweenCalls; /* secs */
7b6 a6e f2      unsigned char maxDialAttempts;
```

```
Hex Add
12x 18x offset
7b7 a6f f3      char cellShare   /* K2 ONLY, 0= 1Hzoutput,
                                             1= cellphone -- FUTURE USE */
7b8 a70 f4      char cellOnTime    /* duration */
7b9 a71 f5      char cellWarmupTime
7ba a72 1 f6-f7 short cellStartTime[5]
7bc a74 2 f8-f9
7be a76 3 fa-fb
7c0 a78 4 fc-fd
7c2 a7a 5 fe-ff
7c4 a7c 100-103  char bytepad[4]

7c8 a80 104-105  unsigned short flags;
                  /* BIT0 = 1 to enable auto call out
                     BIT1 = 1 to call out on battery < 12 V
                     BIT2 = 1 to call out on battery charge failed
                     BIT3 = 1 to call out on extreme temperature
                     BIT4 = 1 to call out on event
                     BIT5 = 1 to call out on GPS lock failure */
7ca a82 106-133 char callOutMsg[46];
             };  // end MODEM_RW_PARMS
      };  // end RW_PARMS
};  // end  KW_HEADER


//------------------
/*  12channel header   0x6c4 + 0x134 (308) = 0x7f8 (2040)
    18channel header   0x97c + 0x134 (308) = 0xab0 (2736)  */
//------------------
```

Altus - Rock Instrument Time Structures Defined

Time is stored in the .EVT File Header and each Frame Header as seconds since January 1, 1980 and in milliseconds.

In the File Header two times are specified:

```
unsigned long       stream_ro_parms.startTime;    /* first sample time,
                                                     includes PreEvent Memory */
unsigned int        stream_ro_parms. startTimeMsec;  /* 0..999 */

unsigned long       stream_ro_parms.triggerTime;
unsigned int        stream_ro_parms.triggerTimeMsec;
```

In each Frame Header, the time of that frame of data is stored as:

```
unsigned long       frame_header.blockTime;      /* data frame time */
unsigned short      frame_header.msec;           /* 0..999 */
```

To convert from

```
    unsigned long, time        /* seconds since 1/1/80 */
    unsigned int, millisecond
```

to

```
        int, year
        int, day_of_year
        int, month
        int, day_of_month
        int, hour
        int, minute
        int, second
        int, millisecond
```

use the following algorithm for years between 1980 and 2100 (2100 is a century that is not a leap year). The year 1980 was a leap year. Four year sets of days (4 years = 1 @ 366, 3 @ 365 = 1461 days) can be used in the time conversion.

```
BEGIN:
millisecond = millisecond;                    // copy milliseconds
second  = (int) (time mod 60);                // modulo or remainder function,
                                              // remainder =  seconds
time    = time / 60;                          // time in minutes

minute  = (int) (time mod 60);                // remainder =  minutes
time    = time / 60;                          // time in hours

hour    = (int) (time mod 24);                // remainder = hours
time    = time / 24;                          // time in days

year    = 1980 + (int) ((time / 1461) * 4);  // 1461 days in 4 years, 3 @ 365,
                                              // 2 @ 366
time    = time mod 1461;
If (time >= 366)                              // more than one year
{
  time  = time - 366;                         // first year of set is the leap year
  year  = year + 1;

  year  = year + (int)(time / 365);           // add second or third year of set
  time  = time mod 365;                      // # of days in last year of set
}
day_of_year = (int)(time + 1);               // add 1 since day 1 = Jan 1

if (year is a leapyear)                      // Leap years are divisible by 4 except
                                             //  century  years
                                             // ( (year mod 100 ? 0) AND
                                             //  (year mod 4 = 0) )
                                             // Leap years are century years
                                             //  divisible by 400
                                             // (2000 is a leap year)
                                             // ( (year mod 100 = 0) AND
                                             //  (year mod 400 = 0) )
{
  if (day_of_year = 60)
  {
     month = 2;
     day_of_month = 29;
     END;
  }

  if (day_of_year > 60)                      // January 31 + Feburary
                                             //  29 = 60 days
    day_of_year = day_of_year - 1;           // adjust so regular routine below
                                             //  will work.
}

int mon[] = {31,28,31,30,31,30,31,31,30,31,30,31};  //  January = 1
int I;

for (I = 1; I <= 12; I++)
{
  if (day_of_year <= mon[i])
  {
     day_of_month = day_of_year;
     month = I;
     END;
  }
  day_of_year = day_of_year - mon[i];
}
```

KW2ASC

The file KW2ASC.SRC, found in the accompaning support software package, is the source code for the program KW2ASC. The KW2ASC program translates the binary data in an .EVT file to ASCII in Volts, a separate file for each channel. See KW2ASC.txt for examples of use and output. The program uses the functions CK2Time(), Seconds2Time(), Day2Month() and IsLeapYear() to translate the .EVT file times. This is a DOS Windows program written in ‘C’ and compiled with Borland v4.52 ‘C++’.