PETLINK™ Walkthrough: mCT 32-Bit Bin-Address Packets 10-Oct-2012 wfj Rev 1.

The purpose of this document is to provide an introductory walkthrough of one aspect of the PETLINK™ Guideline. Here we present examples of various and representative bit fields as they might be used – fields which are currently active in support of the Siemens mCT 32-bit bin-address packet format. The goal is to help the first-time user gain a better, general understanding of the packet format in use.

A companion example 32-bit list-mode file is provided: **ea_32ba_walkthrough.l32**
Two companion example C-code files are provided:

ml_ea_32ba_walkthrough_1.c **[Generates the example *.l32 file.]**
lmsw32ba_verbose_1.c **[Reports contents of *.l32 file verbosely.]**

See also the PETLINK Guideline file.

Overview of list-mode file content:

The list-mode file generated shows examples of various packets with fields being incremented or decremented. Three types of tag packets are used – i.e. elapsed time, horizontal bed position, and lost event tally. [The elapsed time and lost event tally packets are incrementing except for the last maximum value packet examples. The horiz. bed position packets are decrementing except for the last, most-negative value packet.] Two types of event packet fields are shown. The 30-bit BA field is incremented up from zero for a few packets and then incremented up to the maximum value for a few packets. The single-bit Prompt field is shown first with Prompt = 0 (Delayed) examples followed by Prompt = 1 (Prompt) examples as the BA field increments.

Appendix 1: Here is the content of the list-mode generating C-code file, ml_ea_32ba_walkthrough_1.c:

```
//file: ml_ea_32ba_walkthrough_1.c
// make list-mode file - software list mode generator
// Eagle 32-bit bin-address packet field walkthrough
// 9-Oct-2012 wfj

#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <stdlib.h>
#include <iostream.h>

FILE *streamt;

main (int argc, char **argv)
{

      static int i,j;
      static int e1, e2;
      static int b1, b2;
      static int let1, let2;
      static unsigned __int64 i1;

      static char *out_file;

      static int qb;
      static int prompt;     // 0 to 1
      static unsigned int ew1;
      static unsigned int ms = 0;           // milliseconds
      static int hbp = 0;                           // horizontal bed position
      static unsigned int lost = 0; // lost event tally quantity
      static unsigned int ba;
```

```
// get a filename from the command line
    if (argc < 2)
     {
       printf ("usage: %s out_file_name\n",argv[0]);
       exit(1);
     }
    else     out_file = argv[1];

      streamt = fopen (out_file, "wb");
      if ( streamt == NULL) {
             printf ("No file opened %s\n",out_file);
             exit (1);
             }

// Load Output File

// Incrementing Elapased Time Tag Packet
      e1 = 0x80000000 |  (ms++ & 0x1fffffff);
      j = fwrite (&e1, sizeof(qb), 1, streamt);

// Decrementing Horizontal Bed Position Tag Packet
      b1 = 0xc4000000 |  (hbp-- & 0xfffff);
      j = fwrite (&b1, sizeof(qb), 1, streamt);

// Incrementing Lost Event Tally Tag Packet - Type 7 (GIM) & NonFunctional as a Loss
Tally
      let1 = 0xbc000000 |  (lost++ & 0xfffff);
      j = fwrite (&let1, sizeof(qb), 1, streamt);

// Packets to Increment Up from Zero a Portion of the ba Field with Prompt bit cleared
(Delayed)
      ba = 0; prompt = 0;
      for (ba=0; ba <= 15; ba++) {
             ew1 = 0x00000000 | ((prompt & 1) << 30) | (ba & 0x3fffffff);
             j = fwrite (&ew1, sizeof(qb), 1, streamt);
             i1 = i1 + 1;
      }

// Incrementing Elapased Time Tag Packet
      e1 = 0x80000000 |  (ms++ & 0x1fffffff);
      j = fwrite (&e1, sizeof(qb), 1, streamt);

// Decrementing Horizontal Bed Position Tag Packet
      b1 = 0xc4000000 |  (hbp-- & 0xfffff);
      j = fwrite (&b1, sizeof(qb), 1, streamt);

// Incrementing Lost Event Tally Tag Packet - Type 7 (GIM) & NonFunctional as a Loss
Tally
      let1 = 0xbc000000 |  (lost++ & 0xfffff);
      j = fwrite (&let1, sizeof(qb), 1, streamt);

// Packets to Increment Up to Maximum a Portion of the ba Field with Prompt bit set
      ba = 0; prompt = 1;
      for (ba=0x3ffffff0; ba <= 0x3fffffff; ba++) {
             ew1 = 0x00000000 | ((prompt & 1) << 30) | (ba & 0x3fffffff);
             j = fwrite (&ew1, sizeof(qb), 1, streamt);
             i1 = i1 + 1;
      }

// Maximum Value for Elapased Time Tag Packet
      ms = 0x1fffffff; // all bits in 29-bit field set to one
      e1 = 0x80000000 |  (ms & 0x1fffffff);
      j = fwrite (&e1, sizeof(qb), 1, streamt);

// Minimum Value (most negative) Horizontal Bed Position Tag Packet
      hbp = 0x80000;
      b1 =  0xc4000000 |  (hbp & 0xfffff);
      j = fwrite (&b1, sizeof(qb), 1, streamt);
```

```
// Maximum Value Lost Event Tally Tag Packet - Type 7 (GIM) & NonFunctional as a Loss
Tally
      lost = 0xfffff; // all bits in 20-bit field set to one
      let1 = 0xbc000000 |  (lost & 0xfffff);
      j = fwrite (&let1, sizeof(qb), 1, streamt);

      printf (" number of 64-bit packets output: %I64d file size: %I64x\n",i1,8*i1);
      fclose (streamt);
      exit(0);
}
```

Appendix 2: Here is the content of the list-mode verbose reporting C-code file, lmsw32ba_verbose_1.c:

```
// file: lmsw32ba_verbose_1.c
// Verbose Listing for 32-Bit Bin-Address Packets

// 8-Oct-2012 wfj

#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <stdlib.h>
#include <iostream.h>

FILE *streami;

main (int argc, char **argv)
{

      static __int64 i,j,k,i1,j1,j2;
      static __int64 i64_i1;

      static char *in_file;

      static int qb;
      static unsigned int ba;
      static int prompt;
      static unsigned int ew1;
      static int pkt;
      static int tag;

      static int ms;         // milliseconds
      static int hbp;              // horizontal bed position value
      static int hbp_se;     // sign-extended horiz. bed position value
      static int lost;       // lost event tally value
      static int tof_se;     // sign-extended tof value

  // get a filename from the command line

      if (argc < 1)  {
             printf ("usage: %s in_file_name\n",argv[0]);
             exit(1);
    }
      else    { in_file = argv[1];}

      streami = fopen (in_file, "rb");
      if ( streami == NULL) {
             printf ("No file opened %s\n",in_file);
             exit (1);
             }

      k = 0; i64_i1 = 0;
```

```
while  ((i = fread (&ew1, sizeof(qb), 1, streami) ) != 0) {

             pkt = pkt + 1;

             i1 = i1 + 1;

             tag = ((ew1>>31)&1)!=0;

             prompt = 0;

             if(!tag) {
                    prompt = (ew1>>30) & 1;
                    ba = ew1 & 0x3fffffff;
                    printf(" EVENT: pkt_cnt: %I64d ew1(h): %8x prompt: %1d ba(h): %8x
ba(d): %10d\n",
                                                                 i1,   ew1,
prompt,     ba,        ba);
             }

             if ( tag){
                    if ( ((ew1 >> 28) & 0xe)  == 8 ) { // Elapsed Time Tag Packet
                           ms = ew1 & 0x1fffffff; // Extract 29-bit millisecond field
                           printf(" TAG32_ElapsedTime:    pkt_cnt: %I64d ew1(h): %8x
ms(h): %7x ms(d): %9d\n",
                                                                                    i1,
            ew1,        ms, ms);
                    }
                    if ( ((ew1 >> 24) & 0xff) == 0xc4 ) { // Horizontal Bed Position
Tag Packet
                           hbp = ew1 & 0xfffff; // Extract 20-bit bed position field
                           hbp_se = hbp; // Assume hbp is Zero
                           if (((hbp >> 19) & 1) == 1) hbp_se = hbp | 0xfff00000; //
Need to Sign Extend
                           printf(" TAG32_HorizBedPos:    pkt_cnt: %I64d ew1(h): %8x
hbp(h):   %6x hbp_se(d): %8d\n",
                                                                                    i1,
             ew1,        hbp,          hbp_se);
                    }
                    if ( ((ew1 >> 24) & 0xfc) == 0xbc ) { // Lost Event Tally Tag
Packet - Type 7 (GIM)
                           lost = ew1 & 0xfffff; // Extract 20-bit lost tally field
                           printf(" TAG32_LostEventTally: pkt_cnt: %I64d ew1(h): %8x
lost(d): %7d\n",
                                                                                    i1,
            ew1,        lost);
                    }
             }
      }

      fclose (streami);

      exit(0);
}
```

Appendix 3: Here is a text report generated by that “verbose” C-code for the example *.l32 file:

```
TAG32_ElapsedTime:    pkt_cnt: 1 ew1(h): 80000000 ms(h):       0 ms(d):         0
TAG32_HorizBedPos:    pkt_cnt: 2 ew1(h): c4000000 hbp(h):        0 hbp_se(d):        0
TAG32_LostEventTally: pkt_cnt: 3 ew1(h): bc000000 lost(d):       0
EVENT: pkt_cnt: 4 ew1(h):        0 prompt: 0 ba(h):        0 ba(d):          0
EVENT: pkt_cnt: 5 ew1(h):        1 prompt: 0 ba(h):        1 ba(d):          1
EVENT: pkt_cnt: 6 ew1(h):        2 prompt: 0 ba(h):        2 ba(d):          2
EVENT: pkt_cnt: 7 ew1(h):        3 prompt: 0 ba(h):        3 ba(d):          3
EVENT: pkt_cnt: 8 ew1(h):        4 prompt: 0 ba(h):        4 ba(d):          4
EVENT: pkt_cnt: 9 ew1(h):        5 prompt: 0 ba(h):        5 ba(d):          5
EVENT: pkt_cnt: 10 ew1(h):        6 prompt: 0 ba(h):        6 ba(d):          6
EVENT: pkt_cnt: 11 ew1(h):        7 prompt: 0 ba(h):        7 ba(d):          7
EVENT: pkt_cnt: 12 ew1(h):        8 prompt: 0 ba(h):        8 ba(d):          8
EVENT: pkt_cnt: 13 ew1(h):        9 prompt: 0 ba(h):        9 ba(d):          9
EVENT: pkt_cnt: 14 ew1(h):        a prompt: 0 ba(h):        a ba(d):         10
EVENT: pkt_cnt: 15 ew1(h):        b prompt: 0 ba(h):        b ba(d):         11
EVENT: pkt_cnt: 16 ew1(h):        c prompt: 0 ba(h):        c ba(d):         12
EVENT: pkt_cnt: 17 ew1(h):        d prompt: 0 ba(h):        d ba(d):         13
EVENT: pkt_cnt: 18 ew1(h):        e prompt: 0 ba(h):        e ba(d):         14
EVENT: pkt_cnt: 19 ew1(h):        f prompt: 0 ba(h):        f ba(d):         15
TAG32_ElapsedTime:    pkt_cnt: 20 ew1(h): 80000001 ms(h):       1 ms(d):         1
TAG32_HorizBedPos:    pkt_cnt: 21 ew1(h): c40fffff hbp(h):    fffff hbp_se(d):       -1
TAG32_LostEventTally: pkt_cnt: 22 ew1(h): bc000001 lost(d):       1
EVENT: pkt_cnt: 23 ew1(h): 7ffffff0 prompt: 1 ba(h): 3ffffff0 ba(d): 1073741808
EVENT: pkt_cnt: 24 ew1(h): 7fffffe1 prompt: 1 ba(h): 3fffffe1 ba(d): 1073741809
EVENT: pkt_cnt: 25 ew1(h): 7fffffd2 prompt: 1 ba(h): 3fffffd2 ba(d): 1073741810
EVENT: pkt_cnt: 26 ew1(h): 7fffffc3 prompt: 1 ba(h): 3fffffc3 ba(d): 1073741811
EVENT: pkt_cnt: 27 ew1(h): 7fffffb4 prompt: 1 ba(h): 3fffffb4 ba(d): 1073741812
EVENT: pkt_cnt: 28 ew1(h): 7fffffa5 prompt: 1 ba(h): 3fffffa5 ba(d): 1073741813
EVENT: pkt_cnt: 29 ew1(h): 7fffff96 prompt: 1 ba(h): 3fffff96 ba(d): 1073741814
EVENT: pkt_cnt: 30 ew1(h): 7fffff87 prompt: 1 ba(h): 3fffff87 ba(d): 1073741815
EVENT: pkt_cnt: 31 ew1(h): 7fffff78 prompt: 1 ba(h): 3fffff78 ba(d): 1073741816
EVENT: pkt_cnt: 32 ew1(h): 7fffff69 prompt: 1 ba(h): 3fffff69 ba(d): 1073741817
EVENT: pkt_cnt: 33 ew1(h): 7ffffe5a prompt: 1 ba(h): 3ffffe5a ba(d): 1073741818
EVENT: pkt_cnt: 34 ew1(h): 7ffffd4b prompt: 1 ba(h): 3ffffd4b ba(d): 1073741819
EVENT: pkt_cnt: 35 ew1(h): 7ffffc3c prompt: 1 ba(h): 3ffffc3c ba(d): 1073741820
EVENT: pkt_cnt: 36 ew1(h): 7ffffb2d prompt: 1 ba(h): 3ffffb2d ba(d): 1073741821
EVENT: pkt_cnt: 37 ew1(h): 7ffffa1e prompt: 1 ba(h): 3ffffa1e ba(d): 1073741822
EVENT: pkt_cnt: 38 ew1(h): 7fffff0f prompt: 1 ba(h): 3fffff0f ba(d): 1073741823
TAG32_ElapsedTime:    pkt_cnt: 39 ew1(h): 9fffffff ms(h): 1fffffff ms(d): 536870911
TAG32_HorizBedPos:    pkt_cnt: 40 ew1(h): c4080000 hbp(h):    80000 hbp_se(d):  -524288
TAG32_LostEventTally: pkt_cnt: 41 ew1(h): bc0fffff lost(d): 1048575
```

Appendix 4: Here is an example of what the V file viewer utility can show of the *.l32 file:

Getting a copy of Fileviewer.exe: http://www.fileviewer.com/

Setting up Fileviewer (Version 12) for 32-bit packet displayed per line:

View>>Hex Mode (selected)
View>>Hex Formats>> Dword (selected)
View>>Hex Formats>> Flip Ends (selected)
View>>Hex Formats>> Set Hex Line Length (4)

What the V utility shows for this *.l32 file:

	03020100
00000000	80000000
00000004	C4000000
00000008	BC000000
0000000C	00000000
00000010	00000001
00000014	00000002
00000018	00000003
0000001C	00000004
00000020	00000005
00000024	00000006
00000028	00000007
0000002C	00000008
00000030	00000009
00000034	0000000A
00000038	0000000B
0000003C	0000000C
00000040	0000000D
00000044	0000000E
00000048	0000000F
0000004C	80000001
00000050	C40FFFFF
00000054	BC000001
00000058	7FFFFFF0
0000005C	7FFFFFF1
00000060	7FFFFFF2
00000064	7FFFFFF3
00000068	7FFFFFF4
0000006C	7FFFFFF5
00000070	7FFFFFF6
00000074	7FFFFFF7
00000078	7FFFFFF8
0000007C	7FFFFFF9
00000080	7FFFFFFA
00000084	7FFFFFFB
00000088	7FFFFFFC
0000008C	7FFFFFFD
00000090	7FFFFFFE
00000094	7FFFFFFF
00000098	9FFFFFFF
0000009C	C4080000
000000A0	BC0FFFFF