

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 64-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 64-bit list-mode file is provided: **ea_64ba_walkthrough.l64**

Two companion example C-code files are provided:

ml_ea_64ba_walkthrough_1.c

[Generates the example *.l64 file.]

lmsw64ba_verbose_1.c

[Reports contents of *.l64 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 40-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_64ba_walkthrough_1.c:

```
//file: ml_ea_64ba_walkthrough_1.c
// make list-mode file - software list mode generator
// Eagle 64-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 il;

    static char *out_file;

    static int qb;
    static int prompt; // 0 to 1
    static unsigned int ew1, ew2;
    static unsigned int ms = 0; // milliseconds
    static int hbp = 0; // horizontal bed position
    static unsigned int lost = 0; // lost event tally quantity
    static unsigned __int64 ba, bapkt;
```

```

// 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 Elapsed Time Tag Packet
e1 = 0x40000000 | (ms          & 0xffff);
e2 = 0x80008000 | ((ms++ >> 16) & 0x3fff);
j = fwrite (&e1, sizeof(qb), 1, streamt);
j = fwrite (&e2, sizeof(qb), 1, streamt);

// Decrementing Horizontal Bed Position Tag Packet
b1 = 0x40000000 | (hbp          & 0xffff);
b2 = 0x8000c400 | ((hbp-- >> 16) & 0xf);
j = fwrite (&b1, sizeof(qb), 1, streamt);
j = fwrite (&b2, sizeof(qb), 1, streamt);

// Incrementing Lost Event Tally Tag Packet - Type 7 (GIM) & NonFunctional as a Loss
Tally
let1 = 0x40000000 | (lost          & 0xffff);
let2 = 0x8000bc00 | ((lost++ >> 16) & 0xf);
j = fwrite (&let1, sizeof(qb), 1, streamt);
j = fwrite (&let2, sizeof(qb), 1, streamt);

// Packets to Step through ax Field
ba = 0; prompt = 0;
for (ba=0; ba <= 0xf; ba++) {
    bapkt = 0x8000000000000000 |
            (ba & 0xffff)                | (((ba >> 16) & 0xffff) << 32)
|
            (((ba >> 32) & 0xf) << 16)    | (((ba >> 36) & 0xf) << 48)
|
            (((__int64)prompt) << 62);
    ew1 = (int)( bapkt          & 0xffffffff);
    ew2 = (int)((bapkt >> 32) & 0xffffffff);
    j = fwrite (&ew1, sizeof(qb), 1, streamt);
    j = fwrite (&ew2, sizeof(qb), 1, streamt);
    il = il + 1;
}

// Incrementing Elapsed Time Tag Packet
e1 = 0x40000000 | (ms          & 0xffff);
e2 = 0x80008000 | ((ms++ >> 16) & 0x3fff);
j = fwrite (&e1, sizeof(qb), 1, streamt);
j = fwrite (&e2, sizeof(qb), 1, streamt);

// Decrementing Horizontal Bed Position Tag Packet
b1 = 0x40000000 | (hbp          & 0xffff);
b2 = 0x8000c400 | ((hbp-- >> 16) & 0xf);
j = fwrite (&b1, sizeof(qb), 1, streamt);
j = fwrite (&b2, sizeof(qb), 1, streamt);

// Incrementing Lost Event Tally Tag Packet - Type 7 (GIM) & NonFunctional as a Loss
Tally
let1 = 0x40000000 | (lost          & 0xffff);
let2 = 0x8000bc00 | ((lost++ >> 16) & 0xf);
j = fwrite (&let1, sizeof(qb), 1, streamt);
j = fwrite (&let2, 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=0xfffffffff0; ba <= 0xfffffffffff; ba++) {
    bapkt = 0x8000000000000000 |
        (ba & 0xffff) | (((ba >> 16) & 0xffff) << 32)
    |
        (((ba >> 32) & 0xf) << 16) | (((ba >> 36) & 0xf) << 48)
    |
        (((__int64)prompt) << 62);
    ew1 = (int)( bapkt & 0xffffffff);
    ew2 = (int)((bapkt >> 32) & 0xffffffff);
    j = fwrite (&ew1, sizeof(qb), 1, streamt);
    j = fwrite (&ew2, sizeof(qb), 1, streamt);
    i1 = i1 + 1;
}

// Maximum Value for Elapsed Time Tag Packet
ms = 0xffffffff; // all bits in 29-bit field set to one
e1 = 0x40000000 | (ms & 0xffff);
e2 = 0x80008000 | ((ms >> 16) & 0x3fff);
j = fwrite (&e1, sizeof(qb), 1, streamt);
j = fwrite (&e2, sizeof(qb), 1, streamt);

// Minimum Value (most negative) Horizontal Bed Position Tag Packet
hbp = 0x80000;
b1 = 0x40000000 | (hbp & 0xffff);
b2 = 0x8000c400 | ((hbp >> 16) & 0xf);
j = fwrite (&b1, sizeof(qb), 1, streamt);
j = fwrite (&b2, 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 = 0x40000000 | (lost & 0xffff);
let2 = 0x8000bc00 | ((lost >> 16) & 0xf);
j = fwrite (&let1, sizeof(qb), 1, streamt);
j = fwrite (&let2, 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, lmsw64ba_verbose_1.c:

```
// file: lmsw64ba_verbose_1.c
// Verbose Listing for 64-Bit Bin-Address Packets

// 9-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;
    static __int64 i64_i1;

    static char *in_file;

    static int qb;
    static unsigned __int64 ba;
    static int prompt;
    static unsigned int ew1, ew2;
    static int sync;
    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);
    }

    i64_i1 = 0;
```

```

while ((i = fread (&ew1, sizeof(qb), 1, streami) ) != 0) {
    if ((j = fread (&ew2, sizeof(qb), 1, streami) ) != 0) {
        sync = (!((ew1>>31) & 1)) && ((ew2>>31) & 1);

        if (!sync) {
            printf(" PACKET SYNC ERROR 64_bit_word_cnt: %I64d ew1 ew2:
%x %x\n",i64_i1,ew1,ew2);
            exit (1);
        }

        i64_i1 = i64_i1 + 1;

        tag = ((ew1>>30)&1)!=0;

        prompt = 0;

        if(!tag) {
            prompt = (ew2>>30) & 1;
            ba = (((__int64)ew1) & 0xffff) |
            (((__int64)ew2) & 0xffff) << 16) |
            (((__int64)ew1) >> 16) & 0xf) << 32) |
            (((__int64)ew2) >> 16) & 0xf) << 36);
            printf(" EVENT: pkt_cnt: %I64d ew2 ew1(h): %8x %8x prompt:
%1d ba(h): %10I64x ba(d): %13I64d\n",
                i64_i1, ew2,ew1,
                prompt, ba, ba);
        }

        if ( tag){
            if ( ((ew2 >> 12) & 0xe ) == 0x8 ) { // Elapsed Time Tag
                Packet
                ms = ( ((ew2 & 0x1fff)<<16) | (ew1 & 0xffff) ); //
                Extract 29-bit millisecond field
                printf(" TAG64_ElapsedTime:      pkt_cnt: %I64d ew2
ew1(h): %8x %8x ms(h): %7x ms(d): %9d\n",
                    i64_i1, ew2,ew1, ms, ms);
            }
            if ( ((ew2 >> 8) & 0xff) == 0xc4) { // Horizontal Bed
                Position Tag Packet
                hbp = ( ((ew2 & 0xff)<<16) | (ew1 & 0xffff) ); //
                Extract 20-bit bed position field
                hbp_se = hbp; // Assume hbp is Zero
                if (((hbp >> 19) & 1) == 1) hbp_se = hbp |
                0xffff0000; // Need to Sign Extend
                printf(" TAG64_HorizBedPos:      pkt_cnt: %I64d ew2
ew1(h): %8x %8x hbp(h): %6x hbp_se(d): %8d\n",
                    i64_i1, ew2,ew1, hbp, hbp_se);
            }
            if ( ((ew2 >> 8) & 0xfc) == 0xbc) { // Lost Event Tally
                Tag Packet - Type 7 (GIM)
                lost = ( ((ew2 & 0xff)<<16) | (ew1 & 0xffff) ); //
                Extract 20-bit lost tally field
                printf(" TAG64_LostEventTally: pkt_cnt: %I64d ew2
ew1(h): %8x %8x lost(d): %7d\n",
                    i64_i1, ew2,ew1, lost);
            }
        }
    }
}

fclose (streami);

exit(0);
}

```

Appendix 3: Here is a text report generated by that “verbose” C-code for the example *.l64 file:

```

TAG64_ElapsedTime:      pkt_cnt: 1 ew2 ewl(h): 80008000 40000000 ms(h):          0 ms(d):
0
TAG64_HorizBedPos:      pkt_cnt: 2 ew2 ewl(h): 8000c400 40000000 hbp(h):          0
hbp_se(d):              0
TAG64_LostEventTally:   pkt_cnt: 3 ew2 ewl(h): 8000bc00 40000000 lost(d):          0
EVENT: pkt_cnt: 4 ew2 ewl(h): 80000000          0 prompt: 0 ba(h):          0 ba(d):
0
EVENT: pkt_cnt: 5 ew2 ewl(h): 80000000          1 prompt: 0 ba(h):          1 ba(d):
1
EVENT: pkt_cnt: 6 ew2 ewl(h): 80000000          2 prompt: 0 ba(h):          2 ba(d):
2
EVENT: pkt_cnt: 7 ew2 ewl(h): 80000000          3 prompt: 0 ba(h):          3 ba(d):
3
EVENT: pkt_cnt: 8 ew2 ewl(h): 80000000          4 prompt: 0 ba(h):          4 ba(d):
4
EVENT: pkt_cnt: 9 ew2 ewl(h): 80000000          5 prompt: 0 ba(h):          5 ba(d):
5
EVENT: pkt_cnt: 10 ew2 ewl(h): 80000000         6 prompt: 0 ba(h):          6 ba(d):
6
EVENT: pkt_cnt: 11 ew2 ewl(h): 80000000         7 prompt: 0 ba(h):          7 ba(d):
7
EVENT: pkt_cnt: 12 ew2 ewl(h): 80000000         8 prompt: 0 ba(h):          8 ba(d):
8
EVENT: pkt_cnt: 13 ew2 ewl(h): 80000000         9 prompt: 0 ba(h):          9 ba(d):
9
EVENT: pkt_cnt: 14 ew2 ewl(h): 80000000        a prompt: 0 ba(h):          a ba(d):
10
EVENT: pkt_cnt: 15 ew2 ewl(h): 80000000        b prompt: 0 ba(h):          b ba(d):
11
EVENT: pkt_cnt: 16 ew2 ewl(h): 80000000        c prompt: 0 ba(h):          c ba(d):
12
EVENT: pkt_cnt: 17 ew2 ewl(h): 80000000        d prompt: 0 ba(h):          d ba(d):
13
EVENT: pkt_cnt: 18 ew2 ewl(h): 80000000        e prompt: 0 ba(h):          e ba(d):
14
EVENT: pkt_cnt: 19 ew2 ewl(h): 80000000        f prompt: 0 ba(h):          f ba(d):
15
TAG64_ElapsedTime:      pkt_cnt: 20 ew2 ewl(h): 80008000 40000001 ms(h):          1 ms(d):
1
TAG64_HorizBedPos:      pkt_cnt: 21 ew2 ewl(h): 8000c40f 4000ffff hbp(h):          ffffff
hbp_se(d):              -1
TAG64_LostEventTally:   pkt_cnt: 22 ew2 ewl(h): 8000bc00 40000001 lost(d):          1
EVENT: pkt_cnt: 23 ew2 ewl(h): c00ffffff      ffff0 prompt: 1 ba(h): ffffffff00 ba(d):
1099511627760
EVENT: pkt_cnt: 24 ew2 ewl(h): c00ffffff      ffff1 prompt: 1 ba(h): ffffffff11 ba(d):
1099511627761
EVENT: pkt_cnt: 25 ew2 ewl(h): c00ffffff      ffff2 prompt: 1 ba(h): ffffffff22 ba(d):
1099511627762
EVENT: pkt_cnt: 26 ew2 ewl(h): c00ffffff      ffff3 prompt: 1 ba(h): ffffffff33 ba(d):
1099511627763
EVENT: pkt_cnt: 27 ew2 ewl(h): c00ffffff      ffff4 prompt: 1 ba(h): ffffffff44 ba(d):
1099511627764
EVENT: pkt_cnt: 28 ew2 ewl(h): c00ffffff      ffff5 prompt: 1 ba(h): ffffffff55 ba(d):
1099511627765
EVENT: pkt_cnt: 29 ew2 ewl(h): c00ffffff      ffff6 prompt: 1 ba(h): ffffffff66 ba(d):
1099511627766
EVENT: pkt_cnt: 30 ew2 ewl(h): c00ffffff      ffff7 prompt: 1 ba(h): ffffffff77 ba(d):
1099511627767
EVENT: pkt_cnt: 31 ew2 ewl(h): c00ffffff      ffff8 prompt: 1 ba(h): ffffffff88 ba(d):
1099511627768
EVENT: pkt_cnt: 32 ew2 ewl(h): c00ffffff      ffff9 prompt: 1 ba(h): ffffffff99 ba(d):
1099511627769
EVENT: pkt_cnt: 33 ew2 ewl(h): c00ffffff      ffffa prompt: 1 ba(h): ffffffff0a ba(d):
1099511627770
EVENT: pkt_cnt: 34 ew2 ewl(h): c00ffffff      ffffb prompt: 1 ba(h): ffffffff0b ba(d):
1099511627771
```

EVENT: pkt_cnt: 35 ew2 ewl(h): c00fffff ffffc prompt: 1 ba(h): ffffffff c ba(d):
1099511627772
EVENT: pkt_cnt: 36 ew2 ewl(h): c00fffff ffffd prompt: 1 ba(h): ffffffff d ba(d):
1099511627773
EVENT: pkt_cnt: 37 ew2 ewl(h): c00fffff ffffe prompt: 1 ba(h): ffffffff e ba(d):
1099511627774
EVENT: pkt_cnt: 38 ew2 ewl(h): c00fffff fffff prompt: 1 ba(h): ffffffff f ba(d):
1099511627775
TAG64_ElapsedTime: pkt_cnt: 39 ew2 ewl(h): 80009fff 4000ffff ms(h): 1fffffff ms(d):
536870911
TAG64_HorizBedPos: pkt_cnt: 40 ew2 ewl(h): 8000c408 40000000 hbp(h): 80000
hbp_se(d): -524288
TAG64_LostEventTally: pkt_cnt: 41 ew2 ewl(h): 8000bc0f 4000ffff lost(d): 1048575

Appendix 4: Here is an example of what the V file viewer utility can show of the *.l64 file:

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

Setting up Fileviewer (Version 12) for single 64-bit packet displayed per line:

View>>Hex Mode (selected)
View>>Hex Formats>> Double Dword (selected)
View>>Hex Formats>> Flip Ends (selected)
View>>Hex Formats>> Set Hex Line Length (8)

What the V utility shows for this *.l64 file:

| | 0706050403020100 |
|----------|-------------------|
| 00000000 | 8000800040000000 |
| 00000008 | 8000C40040000000 |
| 00000010 | 8000BC0040000000 |
| 00000018 | 8000000000000000 |
| 00000020 | 8000000000000001 |
| 00000028 | 8000000000000002 |
| 00000030 | 8000000000000003 |
| 00000038 | 8000000000000004 |
| 00000040 | 8000000000000005 |
| 00000048 | 8000000000000006 |
| 00000050 | 8000000000000007 |
| 00000058 | 8000000000000008 |
| 00000060 | 8000000000000009 |
| 00000068 | 800000000000000A |
| 00000070 | 800000000000000B |
| 00000078 | 800000000000000C |
| 00000080 | 800000000000000D |
| 00000088 | 800000000000000E |
| 00000090 | 800000000000000F |
| 00000098 | 8000800040000001 |
| 000000A0 | 8000C40F4000FFFF |
| 000000A8 | 8000BC0040000001 |
| 000000B0 | C00FFFFFF000FFFF0 |
| 000000B8 | C00FFFFFF000FFFF1 |
| 000000C0 | C00FFFFFF000FFFF2 |
| 000000C8 | C00FFFFFF000FFFF3 |
| 000000D0 | C00FFFFFF000FFFF4 |
| 000000D8 | C00FFFFFF000FFFF5 |
| 000000E0 | C00FFFFFF000FFFF6 |
| 000000E8 | C00FFFFFF000FFFF7 |
| 000000F0 | C00FFFFFF000FFFF8 |
| 000000F8 | C00FFFFFF000FFFF9 |
| 00000100 | C00FFFFFF000FFFFA |
| 00000108 | C00FFFFFF000FFFFB |
| 00000110 | C00FFFFFF000FFFFC |
| 00000118 | C00FFFFFF000FFFFD |
| 00000120 | C00FFFFFF000FFFFE |
| 00000128 | C00FFFFFF000FFFFF |
| 00000130 | 80009FFF4000FFFF |
| 00000138 | 8000C40840000000 |
| 00000140 | 8000BC0F4000FFFF |