

**MARS 2001 ODYSSEY
NEUTRON COMPRESSION ALGORITHM
Version 1.0**

11/01/2002

Prepared by:
W.C. Feldman

Table of Contents

1. Introduction.....	3
2. Compression Algorithm.....	3
3. Decompression Algorithm.....	3
4. Neutron Compression/Decompression Function	5

1. INTRODUCTION

The GRS neutron spectrometer uses a lossy compression algorithm to reduce 16-bit data received in the spacecraft's central electronics box from the neutron spectrometer to 8-bit data that is transmitted to earth. Once the data has been received back on Earth an 8-bit to 16-bit decompression algorithm is used to uncompress the data.

2. COMPRESSION ALGORITHM

A 16-bit to 8-bit conversion algorithm is used to compress the GRS neutron spectrometer data.

The algorithm is as follows:

$$N(I,J) = (No(I) + [J * 2R(I)])$$

$$\begin{array}{|c|c|} \hline |7-----4|3-----0| \\ \hline | \quad I \quad | \quad J \quad | \\ \hline \end{array}$$

I = 15

While (No[I] - N > 0) //Locate N within table

I=I - 1

J=N - No[I] >> R[I} // shift difference right by R = difference/2R

IJ = (I << 4) + J

Where:

IJ = 8 bit scalar value

I = integer index into table of No and R (in range of 0 to 15)

J = integer number of intervals (in range of 0 to 15)

R = interval size as a power of 2

No = base value

N = 16 bit scalar value

3. DECOMPRESSION ALGORITHM

An 8-bit to 16-bit conversion algorithm is used to uncompress the GRS neutron spectrometer data once it is received on the ground.

$$\begin{array}{|c|c|} \hline |7-----4|3-----0| \\ \hline | \quad I \quad | \quad J \quad | \\ \hline \end{array}$$

$$N = No[I] + (J * 2^{R[I]}) + (2^{R[I]} - 1)/2$$

N = 16 bit value after decompression
 I = Integer index into the table of No and R
 No = Base of range
 2^R = Interval size

I	No	R	2^R	Worst case % error in range
0	0	0	1	0.0
1	16	0	1	0.0
2	32	1	2	1.6
3	64	2	4	2.3
4	128	3	8	2.7
5	256	4	16	2.9
6	512	4	16	1.5
7	768	4	16	1.0
8	1024	5	32	1.5
9	1536	5	32	1.0
10	2048	6	64	1.5
11	3072	7	128	2.1
12	5120	8	256	2.5
13	9216	9	512	2.8
14	17408	10	1024	2.9
15	33792	11	2048	3.0

Example:

If the eight bit number id AEhex

Then I= Ahex = 10dec

And J= Ehex = 14dec

$$N = \text{No}[I] + (J * 2^{R[I]}) + (2^{R[I]} - 1)/2$$

$$N = \text{No}[10] + (14 * 2^{R[10]}) + (2^{R[10]} - 1)/2$$

$$N = 2048 + (14 * 64) + (64 - 1)/2$$

$$N = 2048 + 896 + 31.5 = 2975$$

Note that 31.5 is truncated to an integer

4. NEUTRON COMPRESSION/DECOMPRESSION FUNCTION

```
/******  
*  
* Function Name : lanl_compress  
*  
* Description : compresses the neutron spectrometer spectra by a factor of 2  
*  
* Input Arguments :  
*     input - an array of 16 bit values  
*     num - size of the array  
*  
* Output Arguments :  
*     output - an array of 8 bit values  
*  
* Return Value :  
*     0 - all went well  
******/
```

```
N_Table[0] = 0;  
N_Table[1] = 16;  
N_Table[2] = 32;  
N_Table[3] = 64;  
N_Table[4] = 128;  
N_Table[5] = 256;  
N_Table[6] = 512;  
N_Table[7] = 768;  
N_Table[8] = 1024;  
N_Table[9] = 1536;  
N_Table[10] = 2048;  
N_Table[11] = 3072;  
N_Table[12] = 5120;  
N_Table[13] = 9216;  
N_Table[14] = 17408;  
N_Table[15] = 33792;  
R_Table[0] = 0;  
R_Table[1] = 0;  
R_Table[2] = 1;  
R_Table[3] = 2;  
R_Table[4] = 3;  
R_Table[5] = 4;  
R_Table[6] = 4;  
R_Table[7] = 4;  
R_Table[8] = 5;  
R_Table[9] = 5;  
R_Table[10] = 6;  
R_Table[11] = 7;  
R_Table[12] = 8;  
R_Table[13] = 9;  
R_Table[14] = 10;  
R_Table[15] = 11;
```

```
byte lanl_compress(ushort *input, byte *output, ushort num)  
{  
    byte    ret = 0;  
    ushort  i;  
    ushort  upper;  
    ushort  lower;  
    for (i = 0; i < num; i++)  
    {  
        upper = 15;  
        while (N_Table[upper] - input[i] > 0)  
        {
```

```
        upper--;  
    }  
    lower = (input[i] - N_Table[upper]) >> R_Table[upper];  
    output[i] = (upper << 4) + lower;  
}  
  
    return ret;  
} /* lanl_compress */
```

The decompression algorithm follows.

```
R2_Table[0] = 1;  
R2_Table[1] = 1;  
R2_Table[2] = 2;  
R2_Table[3] = 4;  
R2_Table[4] = 8;  
R2_Table[5] = 16;  
R2_Table[6] = 16;  
R2_Table[7] = 16;  
R2_Table[8] = 32;  
R2_Table[9] = 32;  
R2_Table[10] = 64;  
R2_Table[11] = 128;  
R2_Table[12] = 256;  
R2_Table[13] = 512;  
R2_Table[14] = 1024;  
R2_Table[15] = 2048;  
  
void decomp_lanl(unsigned char *input, unsigned short *output, int num)  
{  
    int i;  
    int split;  
  
    static int init = 0;  
    if (!init) {  
        lanl_init(); // initialize N_Table and R_Table  
        init = 1;  
    }  
  
    //puts(" decompression\n");  
    for (i = 0; i < num; i++)  
    {  
        split = input[i];  
        output[i] = N_Table[HI_NIBBLE(split)] +  
                    (LO_NIBBLE(split) * R_Table[HI_NIBBLE(split)]) +  
                    (R2_Table[HI_NIBBLE(split)] - 1) / 2;  
    }  
} /* decomp_lanl */
```