Mike's Simple Adaptive Block Compression With Silence Compression ©1994 mycal labs www.mycal.net

To Compress a Sample:

Assume BLOCK_SIZE is initialized to a multiple of 16, good values are 80 to 160.
I'm assuming that the data_in[] array is the incoming sample, and that it is filtered with the /4, /2, /4 filter.
I'm also assuming that the data_in[] array is padded to a multiple of the BLOCK_SIZE with 0x80.
Assume data_len = the length of the data_in[] array padded out to a multiple of BLOCK_SIZE;

Initialize Global State Variables before compressing a sample:

```
Ref_sample = 0x0;
Current_block_pointer=0;
```

Compress, a block at a time until there is no data left:

```
While(current_block_pointer<data_len)
        compress a block(current_block_pointer);
        current_block_pointer= current_block_pointer + BLOCK_SIZE;
Else
        We are done;
```

Compress and store block:

Calculate Average Step Size:

```
Prev=data_in[current_sample_pointer] -128;              // convert sample to signed, just flip msb on
                                                        // a byte, or flip upper byte on word.
Step_size=0;
For(i=1;i<BLOCK_SIZE;i++)
{
        t=(data_in[i+current_block_pointer]-128);
        step_size=step_size+abs(t - prev);
        prev=t;
}
step_size=step_size/(BLOCK_SIZE/2);
```

Store step size.

If(step_size!=0) we need to store block

Store Block:

```
For (i=0;i<BLOCK_SIZE;i++)
{
        t=(data_in[i+current_block_pointer]-128);

        if(t > ref_sample)
        {
                store a 1
                ref_sample=ref_sample + step_size;
        }
```

```
    else
    {
        store a 0
        ref_sample=ref_sample - step_size;
    }
}
```

To Uncompress a Sample:

Assume Post Filtering with /4,/2,/4 filter.
Assume BLOCK_SIZE as above.
Assume cdata[] array is an array of bytes produced by the above routine.

Initialize Global State Variables:

```
Ref_sample = 0x0;
Current_block_pointer=0;
```

Uncompress_Block:

```
While(data_pointer<data_len)
    Uncompress_block(data_pointer);
Else
    Were done;
```

Get Step Size:

```
Step_size = cdata[data_pointer];

Data_pointer = data_pointer +1;

If(step_size == 0)
{
    //replicate silence
    for(i=0; i< BLOCK_SIZE;i++)
        store ref_sample;
}
else
{
    // replicate waveform from data
    For (i=0;i<BLOCK_SIZE;i++)
    {
        if(bit=1)
            ref_sample=ref_sample+step_size;
        else
            ref_sample=ref_sample-step_size;
        store ref_sample;
    }
    data_pointer = data_pointer + (BLOCK_SIZE/8);
}
```

EXAMPLE - compress

This sample does not take in consideration the filter, and it can be assumed that the filter is applied before the compression takes place.

Eleven.wav - for this example we use a block size of 16. The first 16 data bytes from eleven.wav are :

134, 133, 133, 131, 129, 127, 127, 125, 123, 120, 121, 122, 122, 125, 128, 128
2nd 16 bytes are:
122, 118, 120, 124, 132, 140, 142, 139, 131, 123, 118, 120, 127, 133, 138, 139

Converted to signed values are:

6, 5, 5, 3, 1, -1, -1, -3, -5, -8, -7, -6, -6, -3, 0, 0
and
-6, -10, -8, -4, 4, 12, 14, 11, 3, -5, -10, -8, -1, 5 10, 11

First block calculated step size is:

step_size = 3 (step= 28, 28/(block_size/2), 28/(16/2), 28/8)
2nd is :
step_size = 9 (step=79, 79/8)

First Compressed Block is :
0x03 0xC8 0x97

This is Step_size =0x03 plus

Ref_sample=0, step_size=3, 1st value = 6 is greater than Ref_Sample, Store 1
Ref_sample=Ref_Sample + step_size = 3

Ref_sample=3, step_size=3, 2nd value = 5 is greater than Ref_Sample, Store 1
Ref_Sample=Ref_Sample + step_size = 6

Ref_sample=6 step_size=3, 3st value = 5 is less than Ref_Sample, Store 0
Ref_sample=Ref_Sample - step_size = 3

Ref_sample=3, step_size=3, 4th value = 3 is less (or =) than Ref_Sample, Store 0
Ref_Sample=Ref_Sample - step_size = 0

Ref_sample=0, step_size=3, 5st value = 1 is greater than Ref_Sample, Store 1
Ref_sample=Ref_Sample + step_size = 3

Ref_sample=3, step_size=3, 6th value = -1 is less than Ref_Sample, Store 0
Ref_Sample=Ref_Sample - step_size = 0

Ref_sample=0, step_size=3, 7th value = -1 is less than Ref_Sample, Store 0
Ref_sample=Ref_Sample + step_size = -3

Ref_sample=-3, step_size=3, 8th value = -3 is less than (or =) Ref_Sample, Store 0
Ref_Sample=Ref_Sample - step_size = -6

Ref_sample=-6, step_size=3, 9th value = -5 is greater than Ref_Sample, Store 1
Ref_sample=Ref_Sample + step_size = -3

Ref_sample=-3, step_size=3, 10th value = -8 is less than Ref_Sample, Store 0
Ref_Sample=Ref_Sample - step_size = -6

Ref_sample=-6 step_size=3, 11th value = -7 is less than Ref_Sample, Store 0
Ref_sample=Ref_Sample - step_size = -9

Ref_sample=-9, step_size=3, 12th value = -6 is greater than Ref_Sample, Store 1
    Ref_Sample=Ref_Sample + step_size = -6

Ref_sample=-6, step_size=3, 13th value = -6 is less than (or =) than Ref_Sample, Store 0
    Ref_sample=Ref_Sample - step_size = -9

Ref_sample=-9, step_size=3, 14th value = -3 is greater than Ref_Sample, Store 1
    Ref_Sample=Ref_Sample + step_size = -6

Ref_sample=-6, step_size=3, 15th value = 0 is greater than Ref_Sample, Store 1
    Ref_sample=Ref_Sample + step_size = -3

Ref_sample=-3, step_size=3, 16th value = 0 is greater than Ref_Sample, Store 1
    Ref_Sample=Ref_Sample + step_size = 0

Second Compressed Block is :
0x09 0x3D 0x17

This is Step_size =0x09 plus

Ref_Sample is carried over from the last block, (it = 0 in this case)

Ref_sample=0, step_size=9, 17th value = -6 is less than Ref_Sample, Store 0
    Ref_sample=Ref_Sample - step_size = -9

Ref_sample=-9, step_size=9, 18th value = -10 is less than Ref_Sample, Store 0
    Ref_Sample=Ref_Sample - step_size = -18

Ref_sample=-18 step_size=9, 19th value = -8 is greater than Ref_Sample, Store 1
    Ref_sample=Ref_Sample + step_size = -9

Ref_sample=-9, step_size=9, 20th value = -4 is greater than Ref_Sample, Store 1
    Ref_Sample=Ref_Sample + step_size = 0

Ref_sample=0, step_size=9, 21st value = 4 is greater than Ref_Sample, Store 1
    Ref_sample=Ref_Sample + step_size = 9

Ref_sample=9, step_size=9, 22nd value = 12 is greater than Ref_Sample, Store 1
    Ref_Sample=Ref_Sample + step_size = 18

Ref_sample=18, step_size=9, 23rd value = 14 is less than Ref_Sample, Store 0
    Ref_sample=Ref_Sample - step_size = 9

Ref_sample=9, step_size=9, 24th value = 11 is greater than Ref_Sample, Store 1
    Ref_Sample=Ref_Sample + step_size = 18

Ref_sample=18, step_size=9, 25th value = 3 is less than Ref_Sample, Store 0
    Ref_sample=Ref_Sample + step_size = 9

Ref_sample=9, step_size=9, 26th value = -5 is less than Ref_Sample, Store 0
    Ref_Sample=Ref_Sample - step_size = 0

Ref_sample=0 step_size=9, 27th value = -10 is less than Ref_Sample, Store 0
    Ref_sample=Ref_Sample - step_size = -9

Ref_sample=-9, step_size=9, 28th value = -8 is greater than Ref_Sample, Store 1
Ref_Sample=Ref_Sample + step_size = 0

Ref_sample=0, step_size=9, 29th   value = -1 is less than than Ref_Sample, Store 0
Ref_sample=Ref_Sample - step_size = -9

Ref_sample=-9, step_size=9, 30th value = 5 is greater than Ref_Sample, Store 1
Ref_Sample=Ref_Sample + step_size = 0

Ref_sample=0, step_size=9, 31st value = 10 is greater than Ref_Sample, Store 1
Ref_sample=Ref_Sample + step_size = 9

Ref_sample=9, step_size=9, 32nd value = 11 is greater than Ref_Sample, Store 1
Ref_Sample=Ref_Sample + step_size = 18