V2.0 of SZIP includes configuration changes and new API functions.

1. Configuration

The Unix configuration and make now builds shared libraries by default. To build only static libraries, use `--disable-shared` option for configure.

The SZIP library may be used with some license restrictions. The decoder (decompression) is free for any use. The encoder is free for non-commercial use, but may require a license for commercial use.

Please see: http://hdf.ncsa.uiuc.edu/doc_resource/SZIP/Commercial_szip.html

The SZIP library may be compiled with or without the encoder enabled. By default, the library is built with the encoder enabled. The resulting library has the same entry points, with the encoder code included or excluded. When compiled with the encoder disabled, the resulting binary library can be used without license.

The `--disable-encoding` option for configure builds SZIP omitting the encoder.

The SZIP library (libsz.a, etc.) includes a variable, `szip_encoder_status`, which is set to the value “SZIP ENCODER ENABLED” or “SZIP ENCODER DISABLED”. Also, the function `SZ_encoder_enabled()` returns 1 if the encoder is available and 0 if not. These mechanisms should be used by applications to determine if SZIP encoding is available.

1. API and Programming

The SZIP library API has been simplified. The following three functions are used to compress and decompress with SZIP.

Note that users of HDF4 or HDF5 must use the HDF API to use SZIP compression.
Name: SZ_BufftoBuffCompress
Signature:

```c
#include "szlib.h"

int SZ_BufftoBuffCompress(void * dest, size_t * destLen, const void * source, size_t sourceLen, SZ_com_t *param )
```

Purpose:
Compress the data in the source buffer into the destination buffer.

Description:
SZ_BufftoBuffCompress attempts to compress the data in source buffer into dest buffer. If destination buffer is big enough, *destLen is set to the size of the compressed data, and SZ_OK is returned. Otherwise, *destLen is unchanged and SZ_OUTBUFF_FULL is returned. If the SZIP encoder is disabled, SZ_NO_ENCODER_ERROR is returned.

The resulting compressed data is a complete SZIP format data stream.

param is a structure of type `SZ_com_t` with parameters that may control compression.

```c
typedef struct SZ_com_t_s
{
    int options_mask;
    int bits_per_pixel;
    int pixels_per_block;
    int pixels_per_scanline;
} SZ_com_t;
```

The options mask defines the following values (defined in `ricehdf.h`):

- `SZ_ALLOW_K13_OPTION_MASK`         1
- `SZ_CHIP_OPTION_MASK`              2
- `SZ_EC_OPTION_MASK`                4
- `SZ_LSB_OPTION_MASK`               8
- `SZ_MSB_OPTION_MASK`               16
- `SZ_NN_OPTION_MASK`                32
- `SZ_RAW_OPTION_MASK`               128
- `Reserved`                         0x10000-ff0000

The `pixels_per_block` must be an even number from 2-32.
When used with HDF, the `bits_per_pixel` should be the number of bits in the HDF data type and `pixels_per_scanline` will be set according to heuristics based on the SZIP specification.

Parameters:

- `void * dest`  
  OUT: Destination buffer
- `size_t * destLen`  
  IN/OUT: Size of the destination buffer; on return is a length of compressed data if successful
- `const void * source`  
  IN: Source buffer
- `size_t sourceLen`  
  IN: Length of the source buffer in bytes
- `SZ_com_t * param`  
  IN: Structure with parameters to control compression; NULL may be passed for default values.

Returns:

- `SZ_OK` if successful
- `SZ_NO_ENCODER_ERROR` if the encoder is not enabled.
- `SZ_CONFIG_ERROR` if the library has been mis-compiled
- `SZ_PARAM_ERROR` if there is an error in parameters list
- `SZ_MEM_ERROR` if insufficient memory is available
- `SZ_OUTBUF_FULL` if size of compressed data bigger than `*destLen`

Name: `SZ_BufftoBuffDecompress`

Signature:
```
#include "szlib.h"

int SZ_BufftoBuffDecompress(void * dest, size_t * destLen, const void * source, size_t sourceLen, SZ_decom_t *param)
```

Purpose:
Decompress the data in the source buffer into the destination buffer.

Description:
`SZ_BufftoBuffDecompress` attempts to decompress the data in source buffer into dest buffer. If destination buffer is big enough, `*destLen` is set to the size of the uncompressed data, and `SZ_OK` is returned. Otherwise, `*destLen` is unchanged and `SZ_OUTBUFF_FULL` is returned. It is assumed that source holds complete compressed SZIP data stream.
param is a structure of type $SZ\_decom\_t$ with parameters that may control decompression. This should be set to the same values used to compress the data. See $SZ\_BufftoBuffCompress$.

**Parameters:**

- `void * dest`
  - OUT: Destination buffer
- `size_t * destLen`
  - IN/OUT: Size of the destination buffer; on return is a length of uncompressed data if successful
- `const void * source`
  - IN: Source buffer
- `size_t sourceLen`
  - IN: Length of the source buffer in bytes
- `SZ\_decom\_t * param`
  - IN: Structure with parameters to control decompression; NULL may be passed for default values.

**Returns:**

- `SZ_OK` if successful
- `SZ_PARAM_ERROR` if there is an error in parameters list
- `SZ_MEM_ERROR` if insufficient memory is available
- `SZ_OUTBUF_FULL` if size of uncompressed data bigger than *destLen

---

**Name:** $SZ\_encoder\_enabled$

**Signature:**

```
int $SZ\_encoder\_enabled(\text{void } )$
```

**Purpose:**

Report whether the encoder is enabled.

**Description:**

$SZ\_encoder\_enabled$ determines whether the SZIP encoder is enabled.

**Returns:**

1 if encoding is allowed, 0 otherwise.