MySETIapp

User's guide

V1.2.12.1

November 20, 2023

MySETIapp is an application that was written to help in analyzing and interpreting the bit stream message from the 'A Sign in Space' project. This is a user driven effort and has no official affiliation with Daniela de Paulis, the SETI Institute, or the European Space Agency or any other collaborating agency or institute.

Background

'A Sign in Space'
website, https://asignin.space/

A Sign in Space is an interdisciplinary project by media artist Daniela de Paulis, in collaboration with the SETI Institute, the European Space Agency, the Green Bank Observatory and INAF, the Italian National Institute for Astrophysics. The project consists in transmitting a simulated extraterrestrial message as part of a live performance, using an ESA spacecraft as celestial source. The objective of the project is to involve the world - wide Search for Extraterrestrial Intelligence community, professionals from different fields and the broader public in the reception, decoding and interpretation of the message. This process will require global cooperation, bridging a conversation around the topics of SETI, space research and society, across multiple cultures and fields of expertise, https://www.seti.org/event/sign-space

The message was transmitted from the ESA's ExoMars Trace Gas Orbiter (TGO) on May 24 at 19:16 UTC/12:15 pm PDT.

It was received by three radio telescopes on earth May 24,2023. A group of individuals in the Discord 'A Sign in Space' channel unscrambled the message from the radio telemetry. The message published as Data17.bin is identified as the correctly transcribed bitstream of the message payload given to ESA.

The next step in the problem is the decoding of the payload bitstream into the next level of the message, the sign in space. After that the interpretation of the sign(s) in the embedded messages can commence.

There is a Discord Interpretation Chat. The group is large and has a wide array of talent. The tools that the group is using is quite varied which includes Excel, Photoshop, GIMP, Java, Python and c/c++.

Several people are also using online tools that use file uploads, typically text based. Importing data into Excel is also generally text based.

Photoshop and GIMP can use raw binary files typically 8 bit, 16 bit or 32 bit per element.

None of these tools use bit packed binary input. The first step has been to translate the bitstream message into a format that can be used by the various tools. The next steps involve examination of the bitstream data to solve how to decode it.

This program contains a set of tools that I have used in the examination of the message. It covers several of the basic files that people are using as a starting point along with several basic bitstream and image formatting , reordering, folding, mirroring, rotation, extraction functions. Hopefully this may make it easier for others people to explore the message.

Limitations

This is a 64 bit Windows Desktop application. It primarily operates from the user's selection of the operations they wish to perform using the application's menus. The operations consist of file input(s), parameters, and file output.

Operations on image files generally result in a new image file. A BMP file of that new image file can be generated. Current implementation uses the default external viewer for BMP files to display results after an operation. On Windows 10 and 11 this is Photos.

ISSUES WITH Windows Photos
Recent updates to Windows Photos make it impossible to turn off image enhancement. This causes blurring of magnified image which makes looking at original pixels difficult. The older Photos is now called Photos Lagacy. This is available for installation through the Windows Store.

V1.0.0.1 Release

This is the first release. It includes the Visual Studio project files along with some of the 'message' data files. An application installer can also be used just for the executable application and data files.

The executable and data files can be stored in any writeable folder in Windows.

Any installer and/or executable should be scanned for viruses when being downloaded along with a verification that the checksum for the installation matches the download.

The installer will add a desktop shortcut to run the application along with a Window Start menu entry named MySETIapp.

There is configuration file, MySETIapp.ini, that will be created when you run the software that keeps track of what files, parameters, window size and positions are used so that you do not have to start from scratch every time you run the application. This is stored in the same folder as the executable. The application does not need to access, record, or transmit information to the Internet.

The settings are only saved when you press the OK dialog button or when the main window and image window display are closed.

V1.1.0.1 Release

This release added:

- Image decimation using 1D or 2D decimation kernels.
- An image file resize function was also added to allow conversion of image file to a different size along with changing the pixel size.
- An external viewer was added to see the results of an image operation.
- Defaults extensions and file type was added to the Open/Save file dialogs.

V1.2.0.1 Release

This release added:

- Added Convert text to packed bitstream file.
- Added image stats reporting not just image header stats.

Corrected error handling of reorder list, when entry in kernel is out of bounds, file closure on error.
Corrected error handling of reorder list, when entry in kernel is out of bounds, file closure on error.
Clean up of ImageDlg to just rely on external viewer. Use of Windows default viewer for BMP display works adequately.
Added batch processing for reordering, this allows a series of reorder kernels to be used. Each kernel adds an index number onto the output filename. An option to generate a BMP file for each is also included.

V1.2.1.1 Release

This release:

Added import BMP to image file (.raw). BMP file must be of type 1 bit, 8 bit, or 24 bit.
Added HEX text file to binary file.
Correction, when display results is enabled only display the results file once.
Correction, Decimation process, corrected incorrect calculation for new Y dimension when row decimation occurs.

V1.2.2.1 Release

This release:

Correction, check that the path exists when specifying file is open/save dialog
Correction, change default folders\filenames
Correction, change default app settings, removed Autosize setting
Corrected bug in V1.2.1 when importing BMP file which resulted in incorrect frame size.

V1.2.3.1 Release skipped

This release was skipped because of a mix-up in the V1.2.2, V1.2.1 release.

V1.2.4.1 Release

This release:

Added, Import of CamIRa IMG files (not applicable to A Sign in Space project)
Added, Add/Subtract constant from images
Added, batch mode for Bit stream to Image file
Changed, Sum images to Add/Subtract images
Cleanup of file open/save dialog filename handling

V1.2.5.1 Release

This release:

Added standard image decimation (summation)
Updated, handling of default directories
Updated, Tab order and default buttons in all dialogs

V1.2.6.1 Release

This release:

Added, Increase image size by replication of pixels.

Added, algorithm driven reordering, only 5 algorithms initially implemented.
Added, Block symbol/phrase extraction from either 1D or 2D image file.
Changed, HEX file output to include # bytes to skip at the beginning of the input file
Changed, Display Image/BMP file to include 16 and 32 bit image files using a scaled 8bpp BMP file. (This is not applicable to the 'A Sign in Space' project.)
Correction, Display Image/BMP file was not displaying some BMP files.

V1.2.7.1 Release

This release:

Added, Inversion of inputs bits in bitstream when using Bit stream to text and Bit stream to image operations.
Added, Insert/add image into an existing image, to able to recreate 'A Sign in Space' starmap from the 5 sign image or to make up one's own message.
Correction, 2d symbol extraction requires image y size to be divisible by the y symbol size.
Change, Block symbol extraction now allows highlighting phrases.
Change, added skeleton for adding 5 more algorithms to algorithm reordering operation.

V1.2.8.1 Release

This release:

Added input2 image file information to dialog for Append end image operation
Added, Space protocol Packet extraction from a TM SPP stream file.
Added, Export Image to bitstream file
Added 4 new algorithms to reorder using algorithm
    Incremental row shoft with wrap aorund
    n Stripes
    Shift (rotate) Rows
    Shift (rotate) Columns
Changed, Append image end, no longer requires Ysize to be the same unless the frames are being added to the end of the first input image.
Changed, bit sequence report to also include 0s as a sequence.
Changed, Add/subtract constant to be add/subract/multiply/divide constant operation
Changed, Add filesize to most bit stream operation dialogs
Correction, 16 bit image pixels were being treated as signed instead of unsigned in certain circumstances.
Correction, reshource.h files to stop ID value are not duplicated. Duplication of ID numbers is something the resource editor automatically does and requires occasional cleanup to keep unwanted GUI actions and compilation errors from occurring. This included checking that a resource ID does not get assigned 65535, this is reserved for IDC_STATIC as -1 (ID #is 16 bits).

V1.2.9.1 Release

This release:

- Added, Bitstream Dialog, Remove NULL bytes sequences from bit stream file
- Added, Image Dialog, Batch processing for reordering using text file with filenames
- Changed, Reorder using algorithm dialog.
  - Added algorithm to reorder using algorithm, Block output [MxN] reordering
  - Added Invert algorithm option
  - Added parameter P3 (for future use, not used in this release)
- Correction, fixed bug in Extract image, when extracted image extended past vertical end of input image.

V1.2.10.1 Release

This release:

- Added, dialog for Add kernel to image on a kernel based grid
- Changed, Reorder using algorithm dialog.
  - Add algorithm, split image left/right
- Changed, Global settings dialog, added auto generate .PNG file flag whenever .BMP generated.
- Changed, Text to bitstream file, added output byte bit order flag
- Changed, Bitstream to binary image
  - added input byte bit order flag
  - dialog changes to clarify which bit order flag applies to input and output files
- Changed, all pixel reordering dialogs, added invert algorithm flag
- Correction, Pixel reordering operations, fixed error so that image files with pixels size of 2 or 4 are saved correctly.

V1.2.11.1 Release

This release:

- Added, Reorder blocks[MxN] in image using text kernel file

V1.2.12.1 Release

This release:

- Added, Prime Number list generator
- Added, Batch Extract image
- Correction, Fixed extract image when extracting multi frame image files

Installation

The installer is a very simple one. **The default install location is the windows volume drive \MySETIapp.** This is used because the application needs to be able to have read/write access of the folder it is installed. The 'Programs folder' requires administrative privileges to write to it. I use 'C:\MySETIapp' as the installation location. You can change the folder installation of the application using the browse button during the install.

The project source is also available on GitHub. The source project and solution uses Visual Studio 2019 or later. It has not been tested against earlier versions.

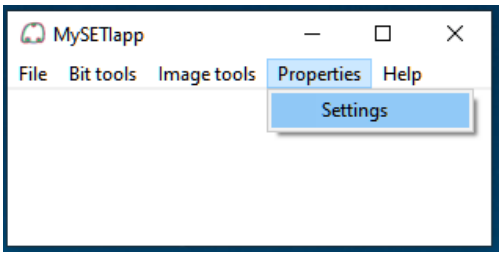
The installer and MySETIapp executable are not digitally signed. You should always check the installation downloads with anti-virus software and verify the MD5 signature of the installer matches just to be safe.

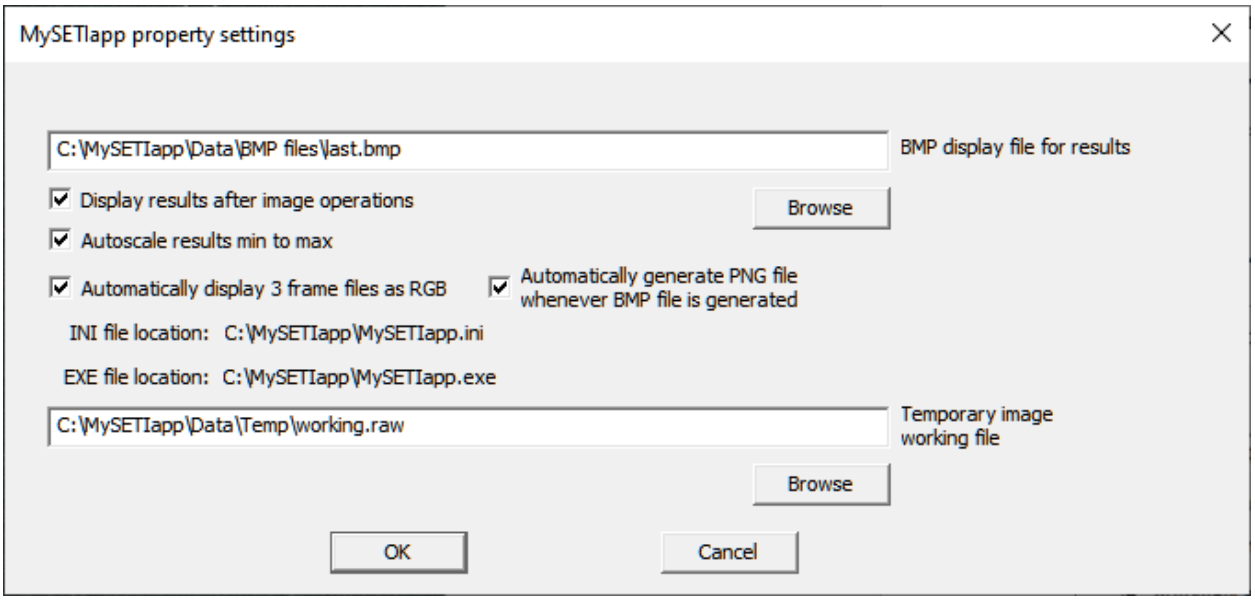
First time running the application

The first time you open a dialog there are defaulted parameters that should be changed and set, in particular the filenames. Use the browse button to select the appropriate folder location for the files. Click OK in the dialog to save the dialogs parameters for the next time. Pressing the OK button saves the dialog settings. It rarely performs the dialog action. Most of the operations in the application have a specific dialog button to press to perform the operation, like Extract, Append, Reorder, …

It is recommended that the application 'settings' be set when you first run the application so that it allows the application to display the results from the last image operation. A BMP file must be specified for this to work.

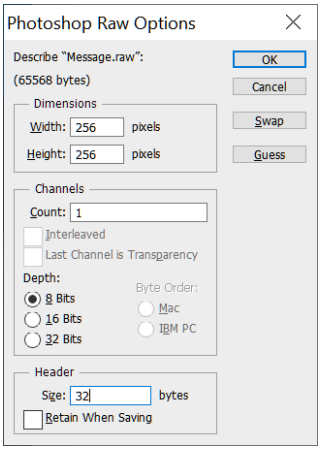
A temporary image working file can be specified. (Currently only used for debugging by saving intermediate processing results.)





IMPORTING IMAGE FILE INTO PHOTOSHOP

Importing an image file into Photoshop is done with files having a .raw extension. In Photoshop do a File -> Open. Select the .raw file you want to import. You will get a Photoshop Raw Options dialog.



Set the header size to 32 bytes. Set the Width and Height to match what is in the raw file. Set the Channels to the number of frame in the image file. Set the Depth to match the image file pixel size. You can get all these parameters from the MySETIapp under the menu Image tools->Image file properties.

IMPORTING IMAGE FILE INTO OTHER APPS

Image files have a 32 byte header. All files are stored in PC endian format (opposite from a MAC). The header structure is:

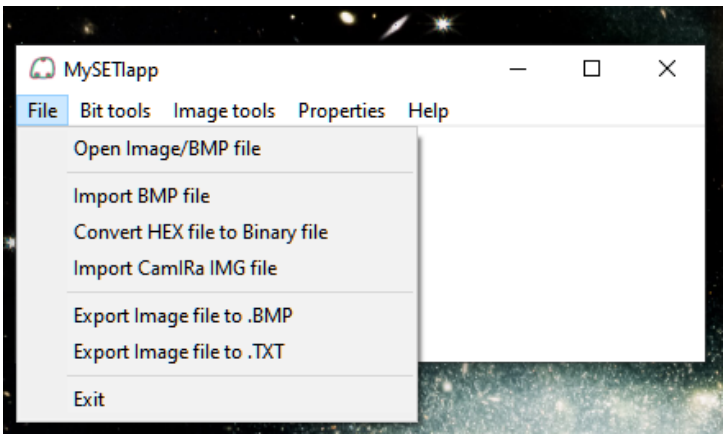
```
#pragma pack(push, 1)
typedef struct IMAGINGHEADER {
        short Endian;           // 0 MAC format, -1 PC format
        short ID;               // 0xaaaa, the header always starts with 0,ID or -1,ID
                                // A file not starting with this is not the correct filetype
        short HeaderSize;       // number of bytes in header
        LONG32 Xsize;           // number of columns in image (type long allows for long linear
bitstreams)
        LONG32 Ysize;           // number of rows image
        short PixelSize;        // pixel size, 1-byte, 2-int16 (short), 4-int32 (int)
        short NumFrames;        // Number of image frames in the file
        short Version;          // header version  number
                                // 1 - this 32 byte header
        short Padding[6];       // dummy entries reserved for other uses
} IMAGINGHEADER;
#pragma pack(pop)
```

Menu operations

Operations are centered around reading and generating files based on parameters entered in the dialog for that operation. The dialog typically needs 1 or 2 input filename, a set of operation parameters and an output filename.

The menus are split into 5 groups with 2 main groups that are comprise the most of the functionality:
File **Bit tools** **Image Tools** Properties Help

File



Open Image/BMP file

This will be implemented in the next release. Currently is just generates the BMP display file for results set in the application properties settings.

Import BMP File

The import BMP file function imports 3 type of BMP file and converts them to an Image file. The BMP must be uncompressed. Color tables in the BMP file are not used or preserved.

Monochrome 1 bit/pixel BMP file

This file type also provides an option to invert the image (0->1, 1->0).

8 bit/pixel BMP file

The image is imported using the image pixel values as a monochrome image. It ignores the color table.

24 bit/pixel BMP file

This type of file is RGB (8 bits per color). This is converted into a 3 frame Image file with Red being the 1st frame, Green being the 2nd frame and Blue being the 3rd frame.

Convert HEX file to binary file

This reads in a text file of HEX ASCII coded bytes which are space delimited, ex: AD 0B 07. Each byte is a 2 character HEX code. Leading 0s are included. This is a straight conversion into a binary file. No interpretation of file type is done.

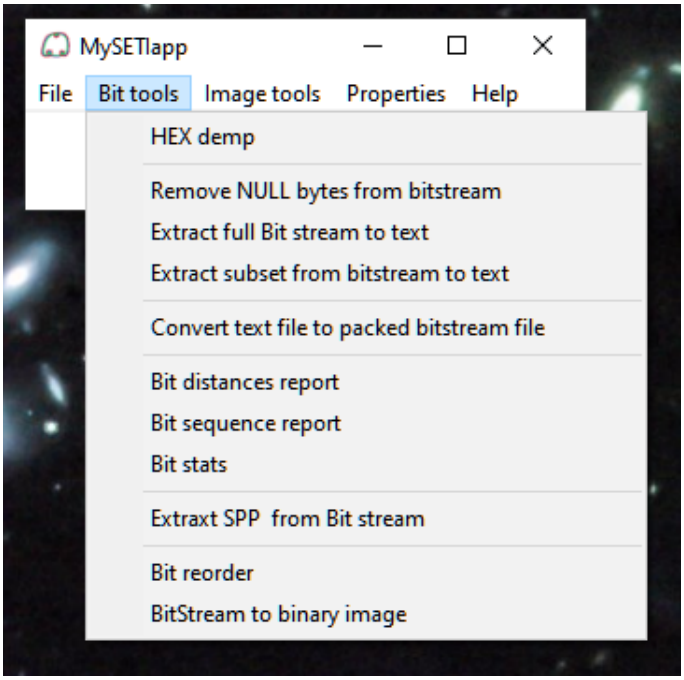
Import CamIRa IMG file

This function imports a CamIRa IMG file into an image file. This is limited to 8 and 16 bit non-extended file formats. No metadata is imported. This function is not applicable to the 'A Sign in Space' project.

Export Image file to .BMP file

Exporting to a BMP file can generate 2 types of BMP files. For image files that are single frame or multi-frame not divisible by 3 the BMP is saved as a greyscale image. For image files with multiples of 3 frames there is an option to use an RGB representation. The first frame is Red, 2nd frame is assigned Green and the third frame is assigned Blue. Only the first frame or 3 frames (RGB output) are converted into a BMP file. You can use auto scale to scale image from black to white. Auto scaling stretches the greyscale bitmap to represent a black to white image even for a single bit pixel.
'Export image file to text' to text allows an image file to be output in a text form that can be used as inputs for programs like Excel or various online applications.

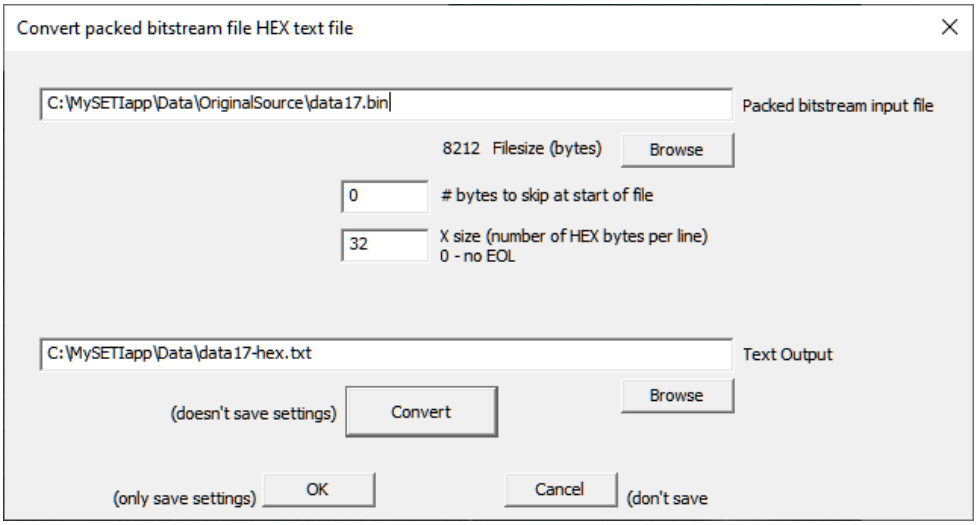
Bit tools



The initial functionality in analyzing the bitstream are the bit tools that allow you to take the packed binary bit stream and examine it. The bit tools can also be used to dismantle and export packed bitstream file in various text or image representations.

It also has some tools that just help you look at what the bitstreams looks like in terms of spacing and sequences.

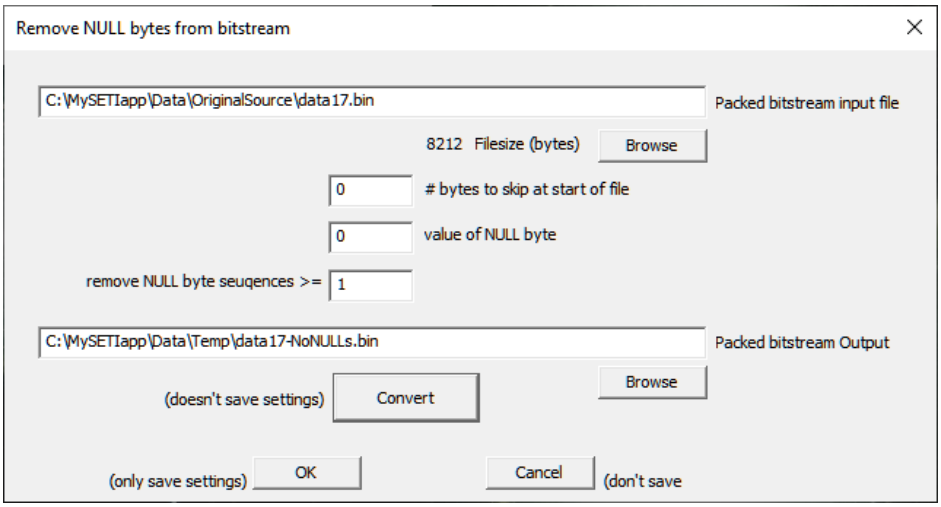
HEX Dump



Hex dump generates a byte oriented hex dump of the input file. This could be either the packed bitstream or and image file. X size is the number of HEX coded bytes to put on a single line. If 0 then all the bytes in the file are output in a single line with no end of line (new line or '\n') at the end. Any file can be HEX dumped. There is white space between the ASCII HEX coded bytes.

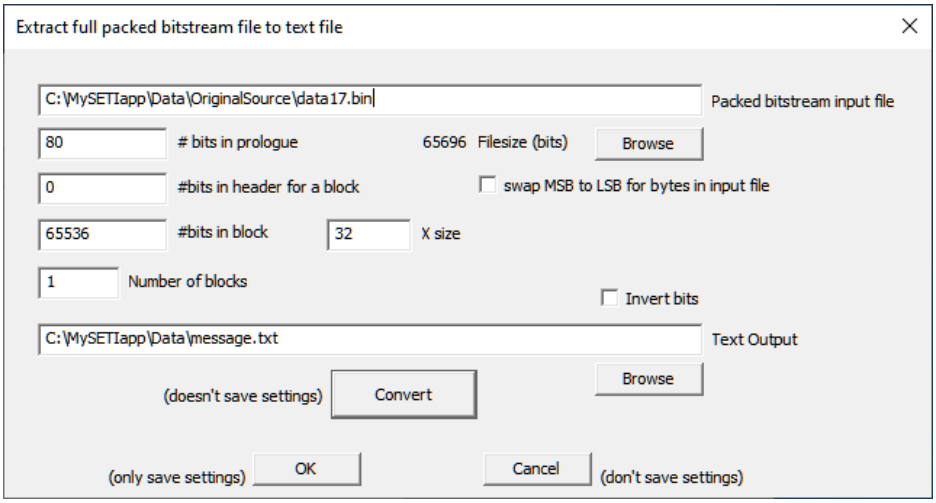
You can specify the number of bytes to skip at the beginning of the input file. This is useful is you do not want to include the 32 byte header an image file.

Remove NULL bytes from bitstream



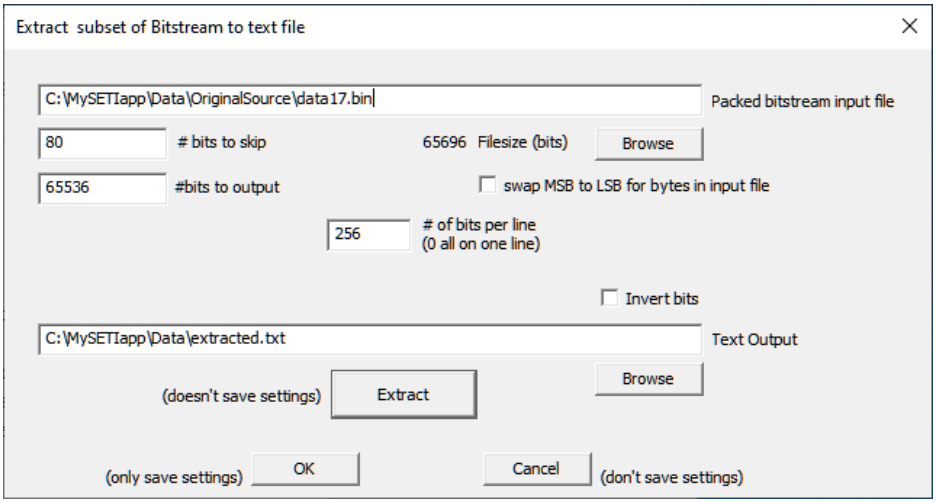
This treats the input bitstream as a stream of bytes, some of which are NULL bytes which should be deleted. This is similar to removing IDLE packet in a protocol. You can specify the # of bytes to skip at the beginning of the file (in case there is a header that needs to be deleted). The value of a byte that is to be treated as a NULL byte. This is likely to be 0. Size of NULL sequence to remove. If 1 remove all NULL bytes. If greater than 1 then delete N-1 NULL bytes in a sequence then start searching again for NULL.

Extract full Bit Stream to text



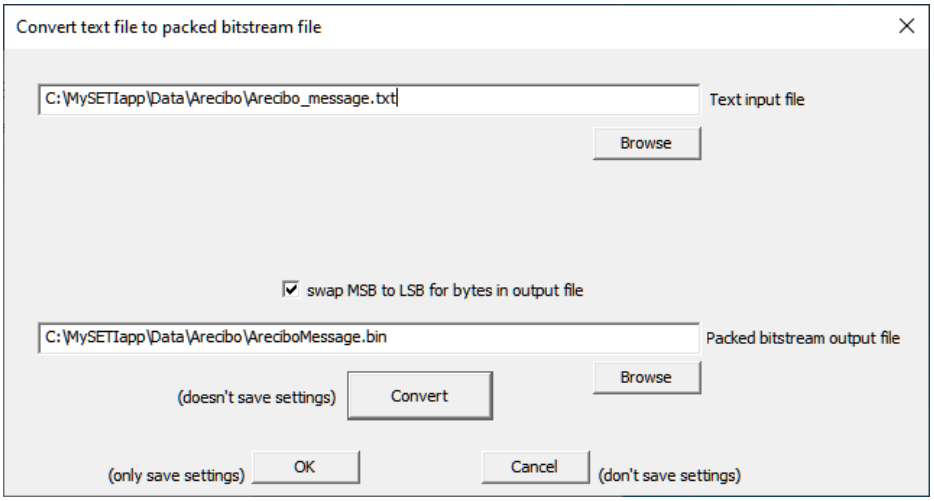
This converts the entire bitstream file into a text file. The text file is separated into 4 sections; prologue(set to 0 to exclude), header bits for a block in a file (set to 0 to exclude), block of bits output in the lines of X size bits, comma separated. The block header and blocks are repeated for the number of block(s) specified, everything after the last block is treated as the footer. The Invert bits flag inverts the input bits from the bit stream. The swap MSB to LSB for bytes in input file reverses the bit order of each byte.

Extract subset from Bit Stream to text



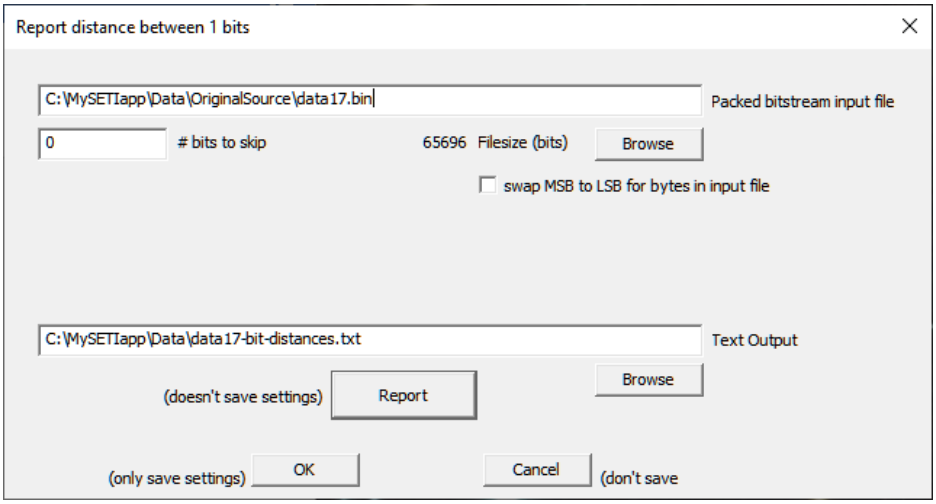
This converts a specified subset bitstream file into a text file. The text file is not separated sections. The will contains lines of X size bits, comma separated. The Invert bits flag inverts the input bits from the bit stream. The swap MSB to LSB for bytes in input file reverses the bit order of each byte.

Convert text file to packed bitstream file



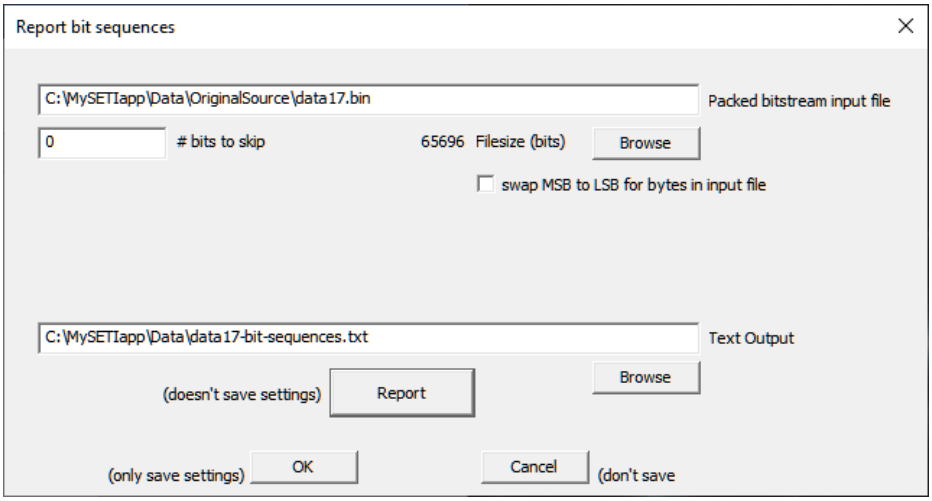
This converts a text file of space delimited values into a packed bitstream file. If a value is less than 0 an error is generated. A value >= 1 is taken as a bit with value of 1. A value of 0 is taken as a bit value of 0. If the number of bits number of values in not a divisible by 8 then 0s are padded to the last byte output in the packed bitstream file. Swap MSB to LSB for bytes in output file reverses the order of bits in a byte.

Bit distances report



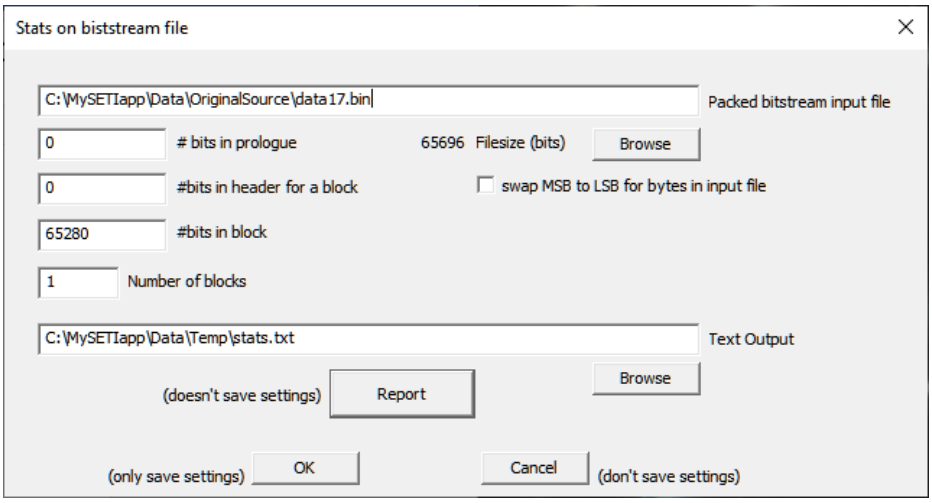
This generates a csv style report of the bit position of each set bit in the bitstream and the distance to the from the last set bit. If the first bit in the file is set the distance is reported as 1. The swap MSB to LSB bot bytes input file reverses the order of bits in input byte.

Bit Sequences report



This generates a csv style report that gives the starting position and length of sequences of set bits. Both 0 and 1 sequences are reported. The swap MSB to LSB bot bytes input file reverses the order of bits in input byte.

Bit Stats



The swap MSB to LSB bot bytes input file reverses the order of bits in input byte.

This generates the following report text file:

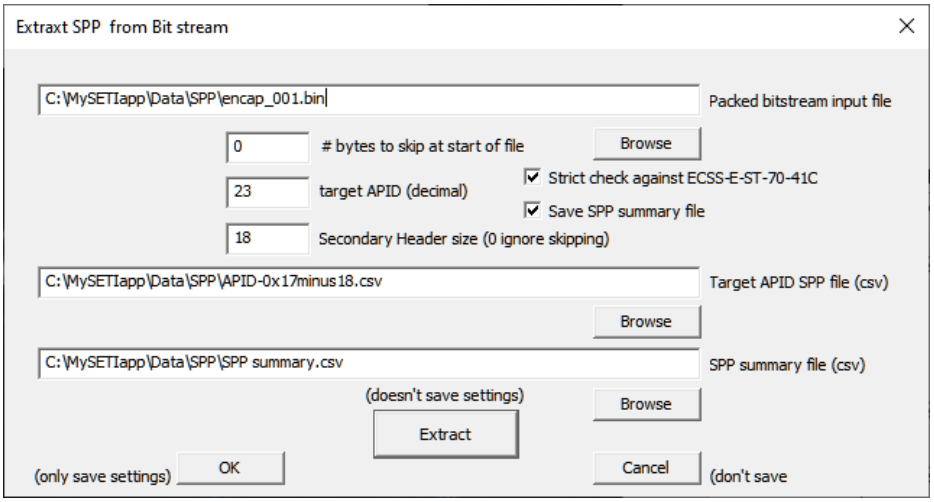
Bitstream file stats
File report settings:
Header size:80
Number of Blocks:1
Header size per block:0
Block size:65536

Bit stats:
Number of bits set in prologue (header): 32, 40.0%

Number of bits set in body, block 0: 625, 1.0%
Number of bits found in footer: 80
Number of bits set in footer: 22, 27.5%
Total number of bits set: 679

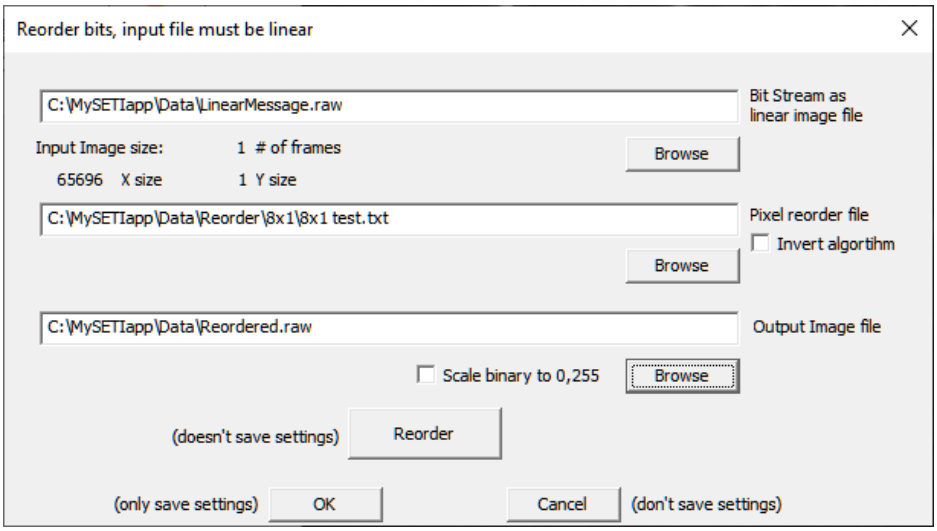
A histogram of the bytes in the bitstream are follow.

Extract SPP from bit stream



The input to this operation is a raw SPP telemetry stream file. This operations looks for the application ID (APID) specified. Note: The APID entered is in decimal not hex. This operation generates 2 files; a file with the full SPP for the target APID that are found, and optionally a summary file containg only the primary header information for all SPPs less the IDLE packets. Check on the formats of a SPP is also done. The check is either strict against ECSS-E-ST-70-4C or loose based upon CCSDS 133.0-B-2. A skip bytes value can be used in cases the SPP telemetry stream file has a header included.

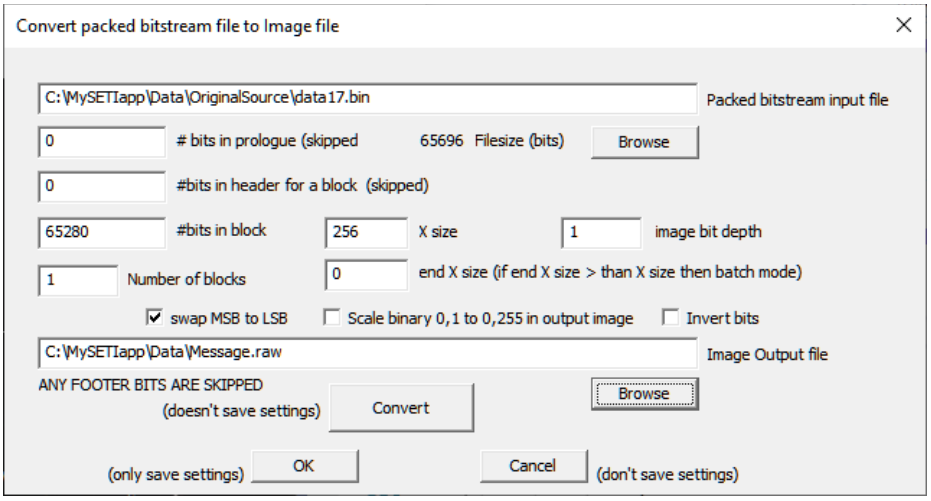
Bit reorder



See the Image reordering documentation for the formats that can be used for the reorder files. When using bit reorder from the bit tools menu the reordering must be 1D (Ysize =1). The X size must be

divisible by the Xsize of the reordering kernel. The linear image is divided into groups the X size of the kernel. The bits within the kernel are reordered according to the kernel specification. The swap MSB to LSB bot bytes input file reverses the order of bits in input byte.

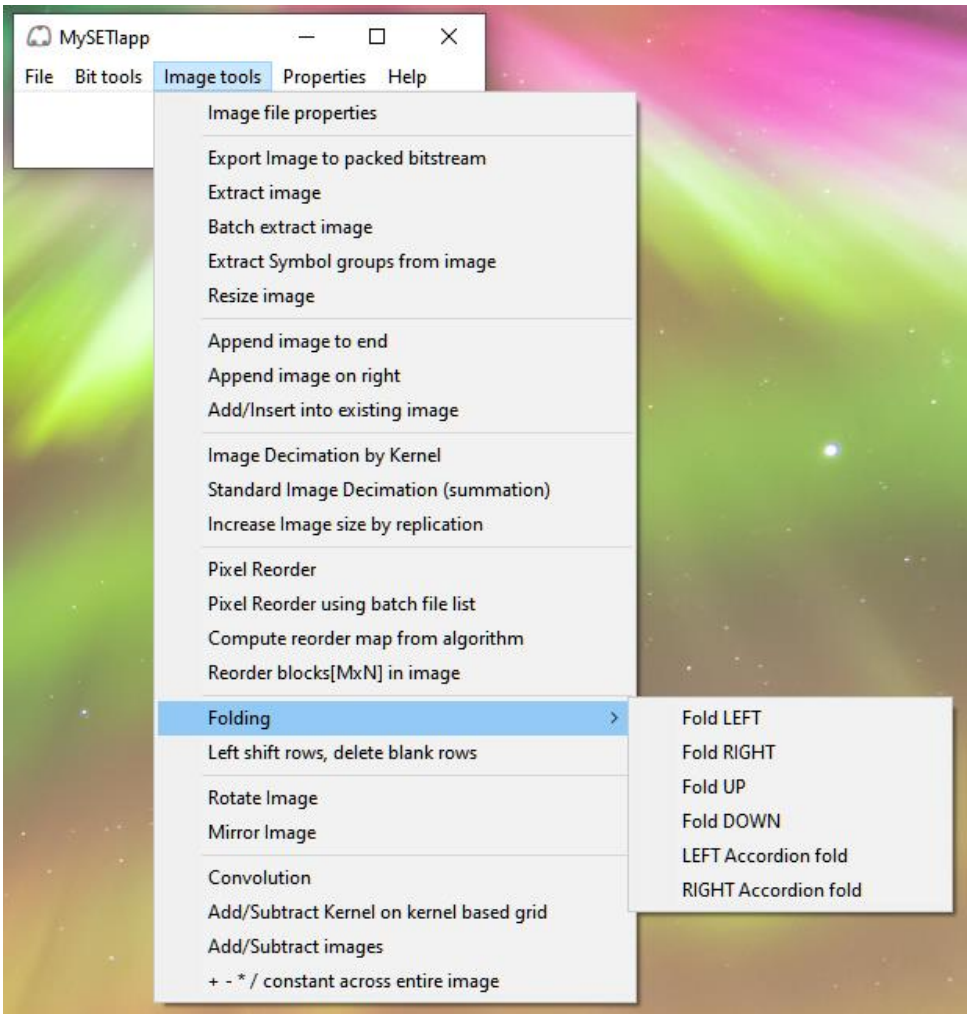
Bitstream to binary image



This does the work of converting packed bitstream files into image files. This includes generating 1D image files which can use 1d linear bit reordering or 1D convolutions on the stream.
The conversion of the bitstream also allows the stream to be treated as having bit depths of 1 to 32 bits. It allows MSB->LSB or LSB->MSB decoding from the bitstream for bit depths > 1. The output can also be scaled so a binary image is displayed from black to white. The # bits in block should be divisible by (Xsize * image bit depth) but is not required. The Ysize of the output image = #bits in block / (X size * image bit depth). The number of blocks becomes the number of frames in the file. . Invert bits flag invert the input bits from the bit stream.

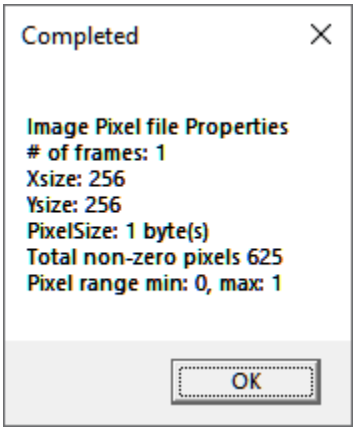
If the end X size is greater than the X size then a batch mode is used. In batch mode the output filename will be appended with an _index matching the X size for each step. If the global Display Results flag is set then a BMP file with the index is also created in the same location but not displayed.

Image Tools



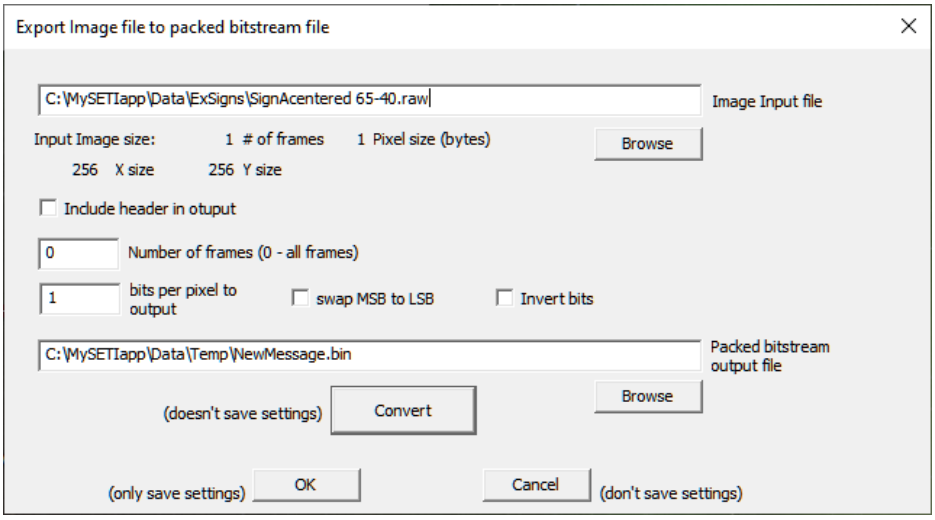
The purpose image tools is to manipulate 2D images. This includes mirroring, rotation, folding, extraction, convolution, summation, and appending images.

Image File Properties



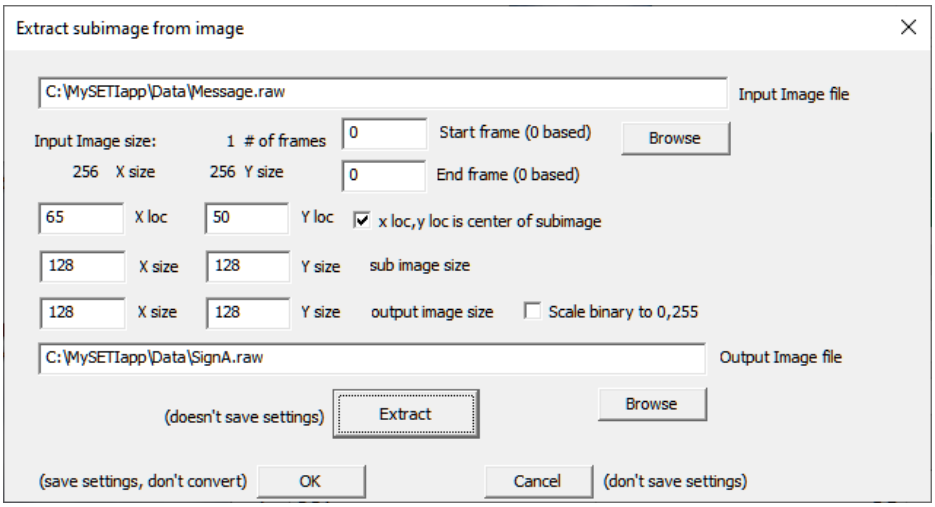
This reports the header information for the selected image file and some basic image stats. It does not generate a report file.

Export Image to packed Bitstream



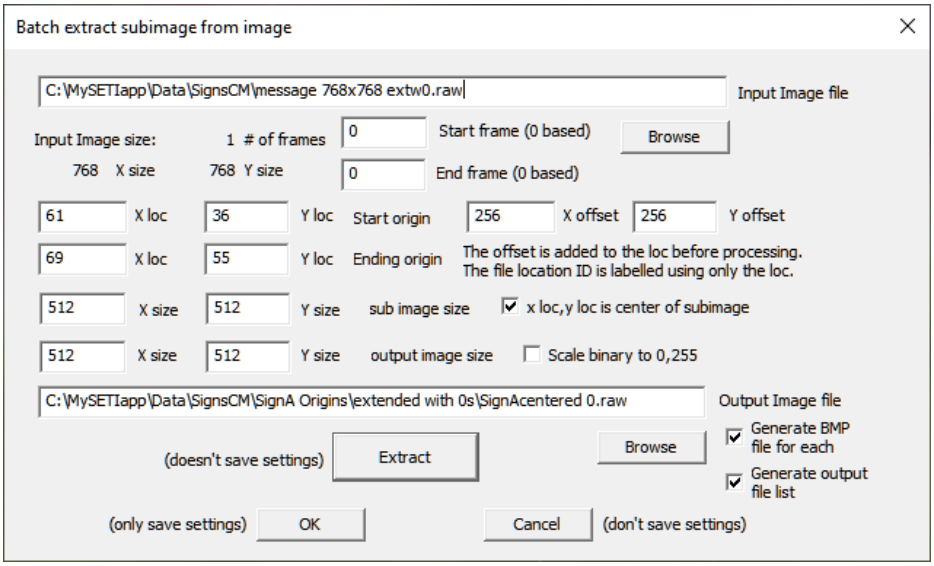
This exports an image file to a packed bitstream. Since image files have bit depths of 8, 16 or 32 bits the bits per pixel to output field is used to select how many bits from a pixel is output to bit stream file. For example: If bit per pixel to output is set to 1 then the image file is treated like it is a single bit image. Invert bits changes the polarity of the output bit(s). Swap MSB to LSB reverses the bit order for a pixel. Note: If bits per pixel to output is set to 1 then bit order will not change. Swapping the bit order also doe not apply to the header if it is included in the bit stream output. See the section at the beginning of this document for the header field definition. The current header is 256 bits.

Extract Image



This allows a sub-image to be extract from a source image. The extraction can be for a range of frames in the source image. The X,Y size of the sub-image must be <= the X,Y size of the output image. The output image can be larger that the input image. The sub-image position is either the upper left corner of the sub-image in the source image or the center of the sub-image in the source image. The output image will be zero padded as needed if the sub-image size extends beyond the bounds of the source image. The use of the centering option can be used to center a specific pixel from the input image to be in the center of the output image. This can make alignments to other sub images easier along with ensuring the other sub images are also the same size.

Batch extract Image



This allows a range of sub-images to be extract from a source image. The extraction can be for a range of frames in the source image. The X,Y size of the sub-image must be <= the X,Y size of the output image. The output image can be larger than the input image. The sub-image position is either the upper left corner of the sub-image in the source image or the center of the sub-image in the source image. The output image will be zero padded as needed if the sub-image size extends beyond the bounds of the source image. The use of the centering option can be used to center a specific pixel from the input image to be in the center of the output image. This can make alignments to other sub images easier along with ensuring the other sub images are also the same size.

The x loc,y loc start origin gives the starting x,y position for the extraction for the first file extraction. The x loc,y loc ending origin gives the last x,y position for the extraction. If the x loc end is less than or equal to x loc start then only the x loc start is used. If the y loc end is less than or equal to y loc start then only the x loc start is used.

The x,y offset is added to the current x,y location. This is useful when than image has been extended with the original image in the center but you still want to use coordinates based on the original image.
For example:
The original image is 256x256 and the x,y location of the sub-image you want to extract centered at 65,40. A new image was generated that is 768x768 in size with the original 256x256 in it center. Then using the offset 256,256 would allow you to still use the 65,40 coordinate for the extraction from the larger image.

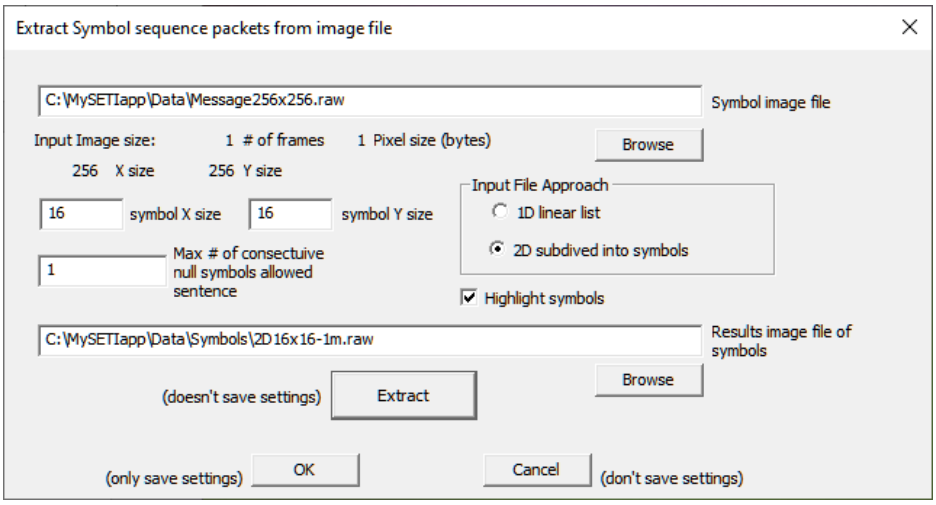
The output image filename specifies the base filename. It is appended with the x-y location for that file. Example: SignA.raw becomes SignA 65-40.raw

If the generate BMP file is checked then a BMP with the same filename as the output filename is also generated.

If the generate file list is also check then 2 more files are generated in the output file directory. The first one is 'FileList.txt' and is the list of all the files generated in this process. The second file is

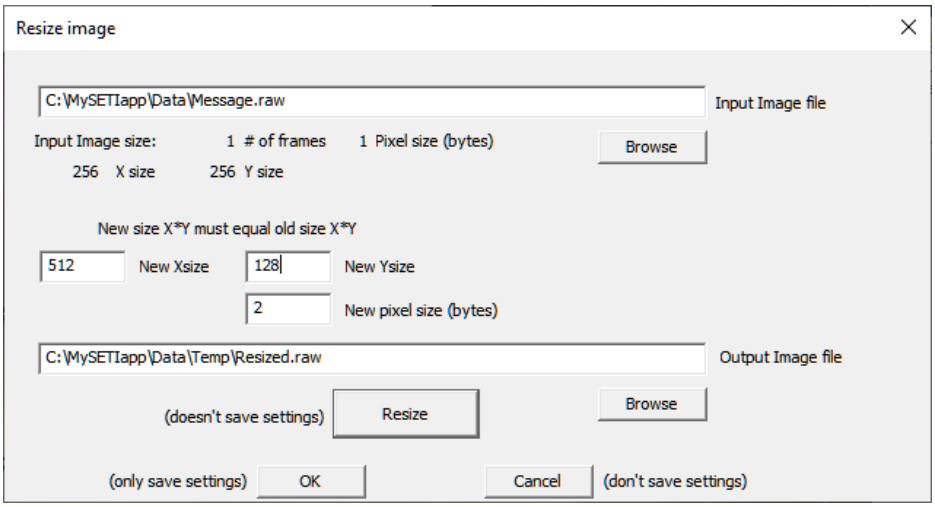
'BatchFileList.txt' It contains the name of the file generated and the name of the file generated with the '\Batch' directory inserted followed by a blank line. This is to facilitate the batch reorder image process.

Extract Symbol sequence packets for image file



This function extracts block symbols from an image file. This can be either a 1D extraction or a 2D extraction. This also creates phrases which are a series of symbol separated by small blank space (null symbols) and ended with a longer blank space (null symbols). Highlight symbols is enabled that the null symbols in the results image will be used to highlight the symbols and phrases.

Resize Image



This operation copies an image file to another image file while changing the X, Y size and the Pixel Size. This is useful for converting a flat image file (Y size is 1) to a 2D image. It also allows the Pixel size to be changed between 1, 2 or 4 bytes per pixel.

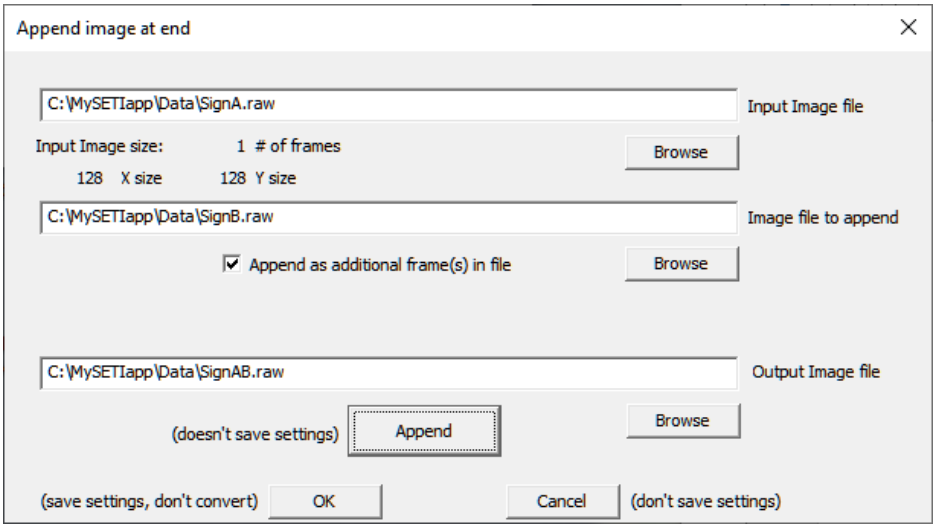
Limitations:

The new image (X size * Y size) must be equal to the old image (X size * Y size). No removal or additional pixels can be added to a the frame. The new image file will have the same number of frames as the old file.

If the new Pixels size is smaller than the old pixel size then the values will be clipped as follows:

1 byte - clipped at 255
2 bytes - clipped at 65535
4 bytes - negative numbers will be set to 0

Append image to end



Images are be appended at the end of another image or added as additional frame(s) in the image file.

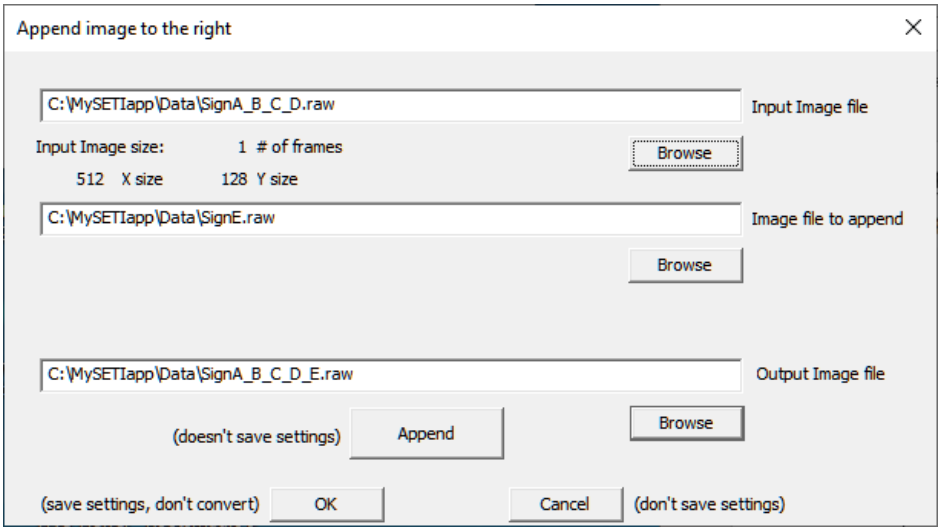
Appended to frame condition

If the 2nd image is being appended on the first frame without being added as an additional frame then the output image will be the Y size of the first image plus the Ysize of the second image. Appending the X size of both input image files be the same along with the same number of frames in each input file.

Appended as additional frames condition

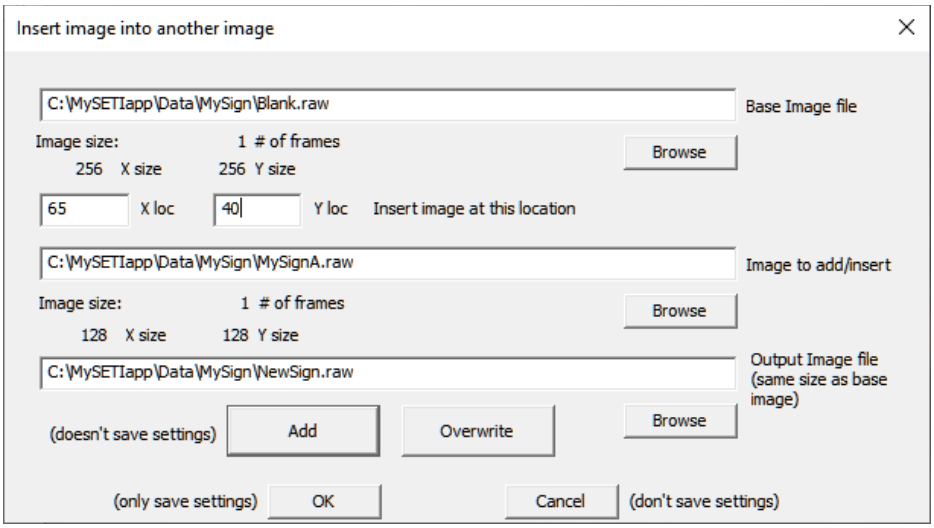
The first and 2nd input image files must have the same X, Y size. The number of frames in each input image file can be different. The number of frames in the output image will be the number of frames from the first input image plus the number of frames from the 2nd input image. This is particularly useful when generating a 3 frame file that you want to convert to an RGB representation. When exporting a 3 frame file the 3 frames can be interpreted as frame 1 is Red, frame 2 is Green and frame 3 is Blue.

Append image on right



This appends the second image to the right side of the first image. This can be used to string a set of sub-images together into one image with a common x and Y alignment. Such as the possible 5 signs in the "A Sign In Space' message.

Add/Insert into existing image



Insert or adds and image into an existing image. Normally the image being added would be smaller than the existing image. The existing image sets the output image size and is not extended. If the image being added/inserted extends outside the bounds of the existing image then those portions of the added/inserted image are excluded. The added/inserted image is centered at the x,y location specified in the existing image. Selecting 'Add' sums the the image into the existing image. Selecting 'Overwrite' replaces that area of the existing image with the inserted image.

Image Decimation by Kernel

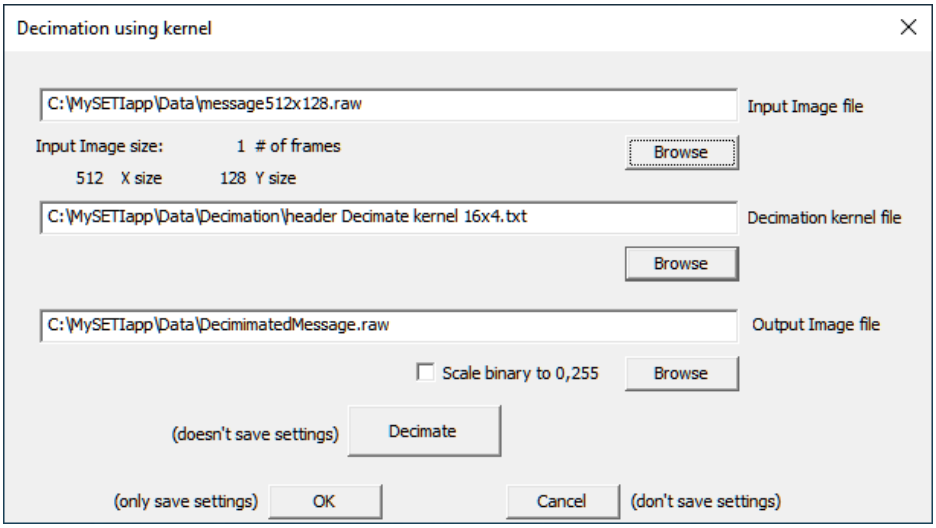


Image decimation reduces the X and Y resolution of an image by deleting pixels in the image. How the pixels are removed is based on a decimation kernel file. You can create decimation kernels that are 1D or 2D. 1D kernels remove pixels in a row or a column but not both. 2D kernels can removed pixels in both rows and columns. The first line of a kernel specifies the x and y size of the kernel and must be comma separated. A list (x*y) of kernel values follow that are separated by whitespace. A kernel value can be 0 or 1. Values of 0 will result in that pixel being removed from the image. A value of 1 includes a pixel in the new image. A description can be added after all the kernel values.

Limitations:

For 2D kernels, the number of decimated pixels in each rows must be the same unless all the pixels on a row are removed (all 0s).

The input image x, y size must both be divisible by the decimation kernel x, y size.

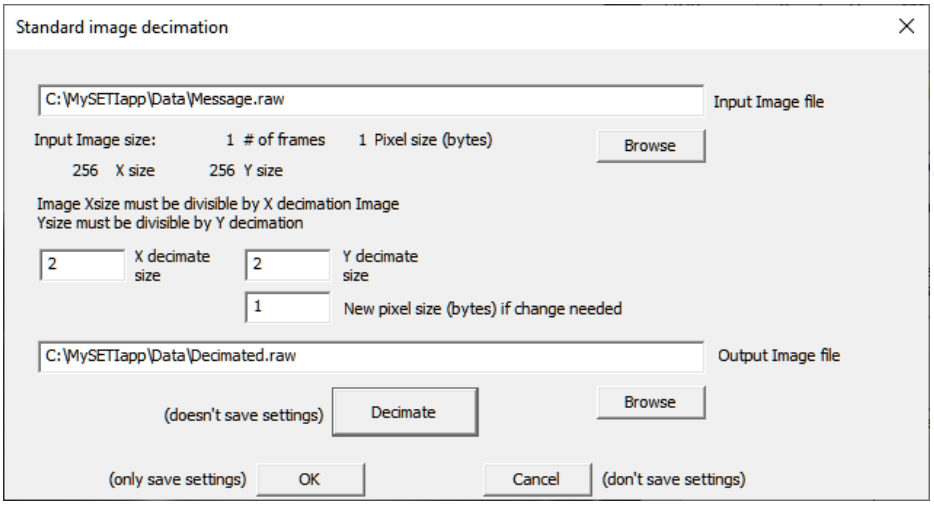
A new size for the output image is calculated from the decimation kernel and the input image size.

Examples:

2,2
0 0
0 1
This removes every even number column and row from the image.

16,4
0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0
0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1
0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0
0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0
16x4 linear decimation, using message header as a decimation kernel
This would change a 512Hx128V image to a 128Hx128V image.

Standard Image Decimation (summation)

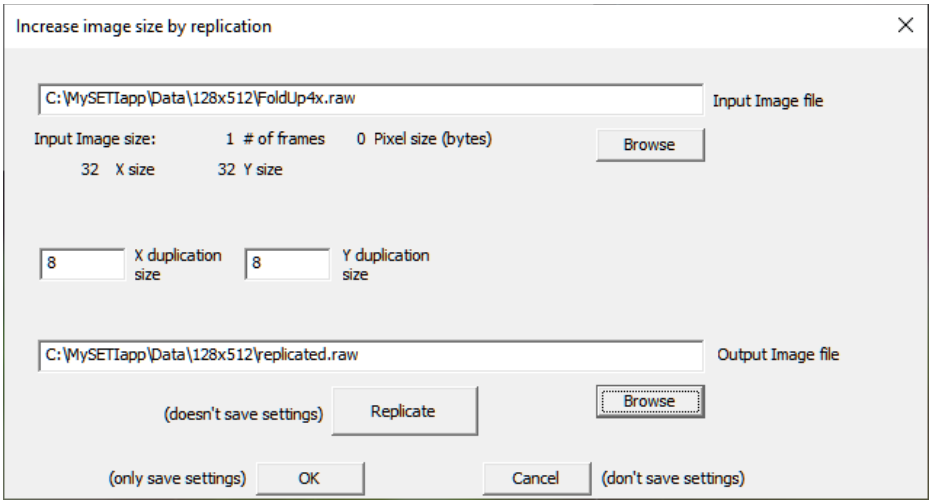


Standard Image decimation reduces the X and Y resolution of an image by summing groups of pixels in the image. The x,y decimation size is the size of the group. All pixels in a group are summed and become a new pixel value in the output image.

Restrictions:

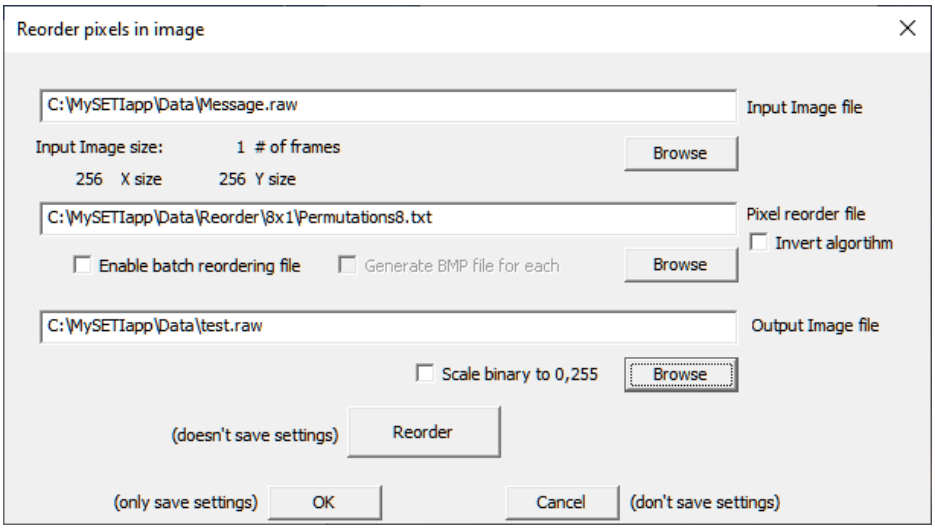
The x,y size of the image must be divisible by the x,y decimation size.

Increase image size by replication



This increases the image size by the x,y the duplication values. The new image size if original x size * x duplication size by original y size * y duplication size. Each pixel in the original image is turned into a block x,y duplication size and written to the new image file.

Pixel Reorder



1D/2D pixel reordering divides an image up into blocks which are the size of the reordering kernel. The pixels inside each block are reordered according to the kernel specification. The limit on the size of the kernel is the X,Y size of the image. This of course would allow you to arbitrarily remap any pixel to any other place in the image. Someone could make the resulting image into almost any representation desired. It is more likely that a much smaller reordering kernel would be used.

Batch processing of kernel is also an option. This option lets the user explore many possible permutation orders for reordering in one request. For example, 8 column reordering has 40320 possible permutations for the order. While it is easy to generate the entire list, it is tedious to perform this one by one. Using the generate BMP option, you can generate a movie from the resulting BMP file using Photoshop or other tool. This lets you watch the results as a video.

When the Enable Batch flag is checked then the reordering file is interpreted as having 1 or more kernels sequentially listed in the reordering file. Batch processing ends when a comment, end of file or an error in a kernel is encountered. An index number is added to the filename starting at 1. If the Generate BMP file flag is also checked then a ,BMP file is also generated with the same name as the .raw file (including the index number).

If the Invert algorithm flag is set then the inverse of the reordering transforms is performed.

File format for reordering files

The reordering file for 1D/2D reordering has the following 3 text formats.

The following example kernels reverses the order of pixels left to right in the block and swaps the rows so that an 8x2 block of pixel values:

```
0 1  2  3  4  5  6  7
8 9 10 11 12 13 14 15
becomes
15 14 13 12 11 10 9 8
 7  6  5  4  3  2 1 0
```

1.) Relative pixel mapping format

n,m

followed by n*m pairs of values with whitespace between the pairs. After all the n*m pairs of values are listed an optional description is recommended. The reordering values are relative to the its position in the kernel. A 0,0 means the pixel is not moved.

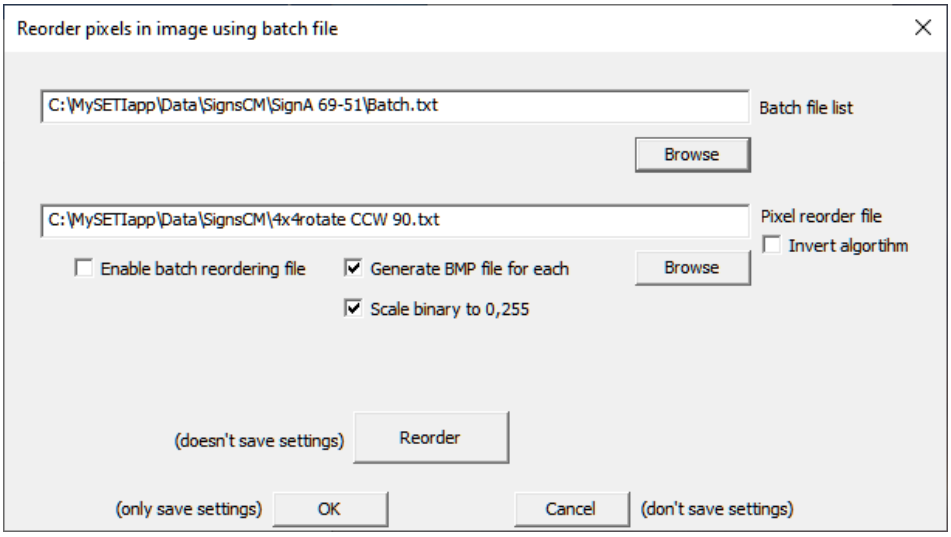
Example file contents format – relative pixel maps
8,2
7,1 5,1 3,1 1,1 -1,1 -3,1 -5,1 -7,1
7,-1 5,-1 3,-1 1,-1 -1,-1 -3,-1 -5,-1 -7,-1
8X2 kernel. This swaps the two rows and reverse the order of a row

2.) 0 based linear kernel address format
n,m,0
followed by n*m values with whitespace between the values after all the n*m values an optional description is recommended. Reordering values is the linear address of replacement pixel in the kernel, 0 based.

Example file contents format –0 based linear kernel address
8,2,0
15 14 13 12 11 10 9 8
7 6 5 4 3 2 1 0
8X2 kernel. This swaps the two rows and reverse the order of a row

3.) 1 based linear kernel address format
n,m,1
followed by n*m values with whitespace between the values after all the n*m values an optional description is recommended. Reordering values is the linear address of replacement pixel in the kernel, 1 based.
Example file contents format –1 based linear kernel address
8,2,1
16 15 14 13 12 11 10 9
8 7 6 5 4 3 2 1

Pixel reorder using batch file list



This allows the processing of the reordering operation by a groups of files lists in a text file. This operation is the same as the Pixel Reorder dialog except that the filenames come form a text file. This allows one to process many different images in one step.

If the Invert algorithm flag is set then the inverse of the reordering transforms is performed.

The file format is:
Blanks line and lines starting with / or ; or : are treated as comments and ignored

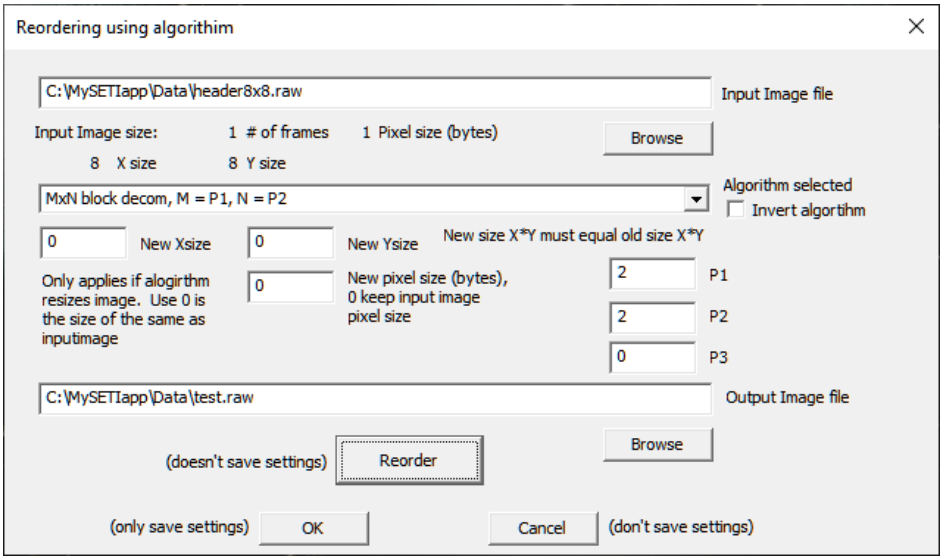
Input image filename
Output image filename
The filename s should be fully qualified for predictable results.

Example file:
C:\MySETIapp\Data\SignsCM\SignA 69-51\SignA -1 -1.raw
C:\MySETIapp\Data\SignsCM\SignA 69-51\Batch\SignA -1 -1 Batch.raw

C:\MySETIapp\Data\SignsCM\SignA 69-51\SignA -1 -2.raw
C:\MySETIapp\Data\SignsCM\SignA 69-51\Batch\SignA -1 -2 Batch.raw

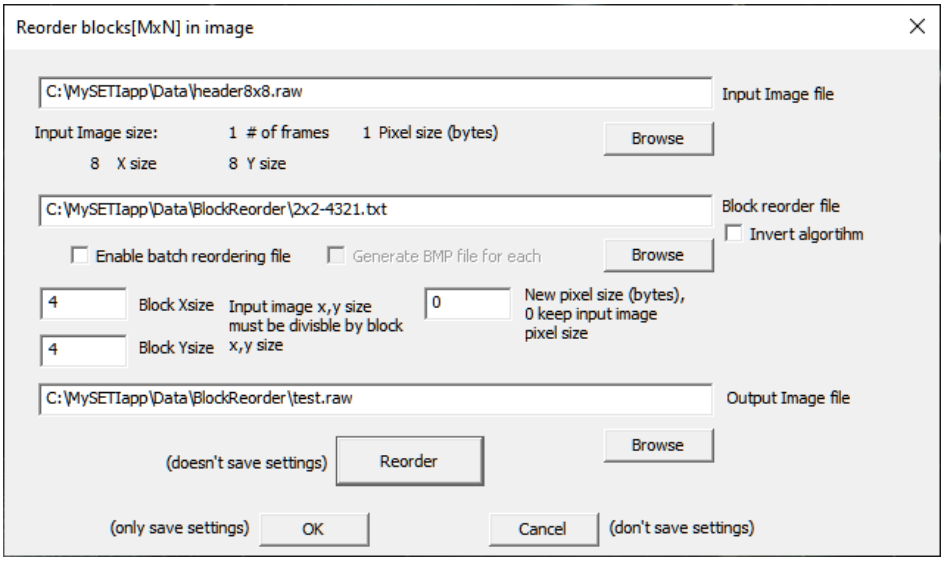
C:\MySETIapp\Data\SignsCM\SignA 69-51\SignA -1 0.raw
C:\MySETIapp\Data\SignsCM\SignA 69-51\Batch\SignA -1 0 Batch.raw

Compute reorder map from algorithm



The image is reordered using the selected algorithm rather than a file based kernel. Some reordering operations would require a kernel the entire size of image but can also be easily performed as an algorithm instead which effectively generate the full size kernel based on parameters. The operator can also select to perform the invert of the algorithm. This allows both forward and backward transformations.

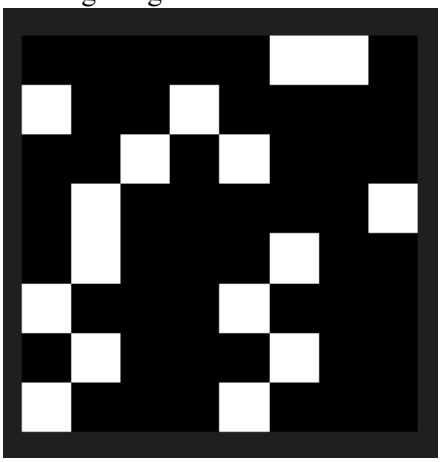
Reorder blocks[MxN] in image



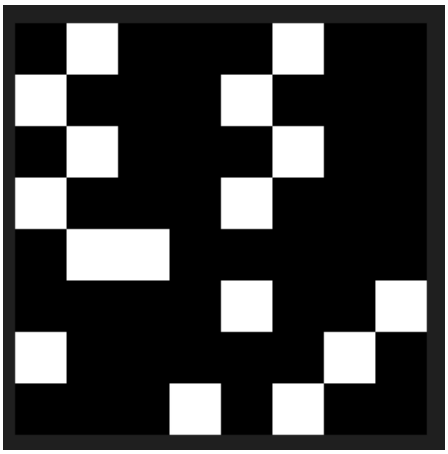
The operation splits the image up in blocks in the image. It moves the blocks in the image based of the order specified in the Block reorder file. This is similar to the pixel reordering specifications which reorders pixel within a block. There are requirements which must be met for this transformation to work.
The image x size must be divisible by the Block x size.
The image y size must be divisible by the Block y size.
The kernel x size must be the image x size divided by the Block x size.
The kernel y size must be the image y size divided by the Block y size.
Example: 256x256 image with a block size of 8x8 required a kernel size of 32x32
If these conditions are not met then an error will be generated.

The Block reorder file format is the same format as the pixel reorder format. See the pixel reorder file specification in that section for details. The typical example would be:
2,2,1
4 3
2 1
This is a 2x2 1based kernel

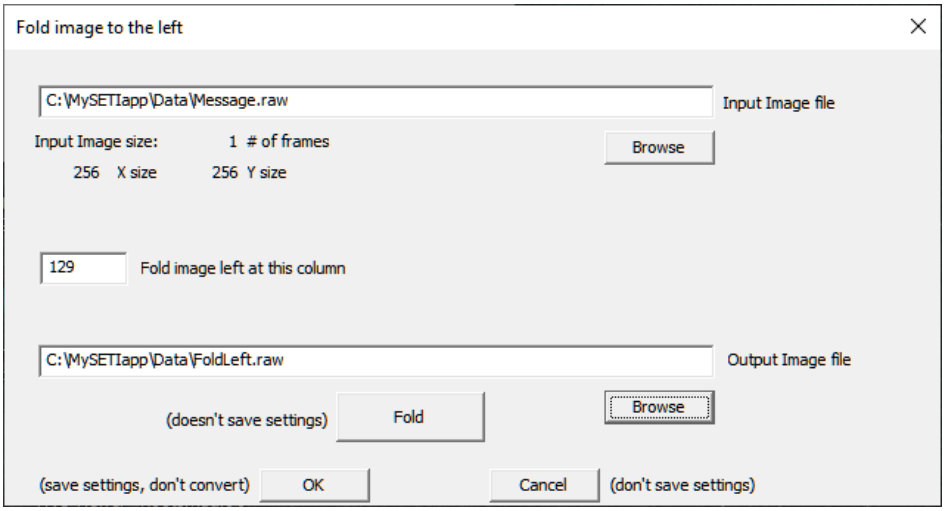
Starting image



Ending image

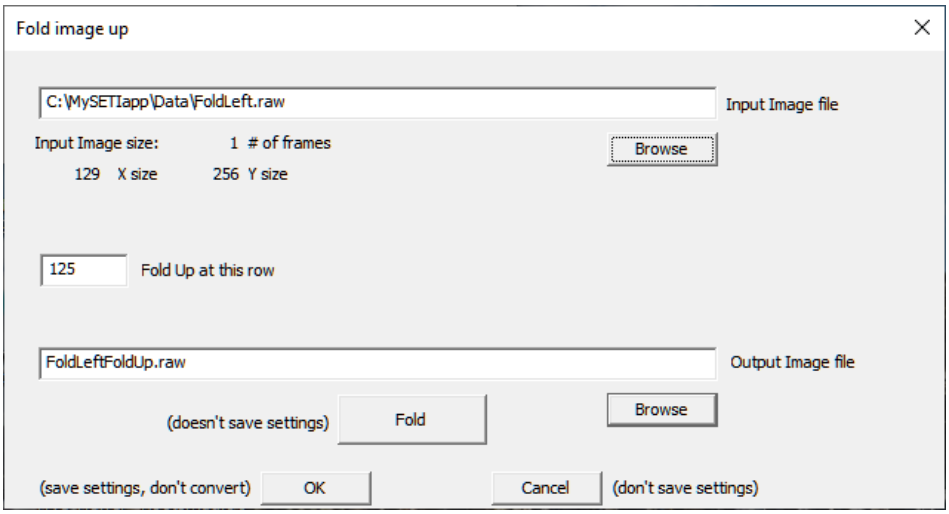


Folding along column (LEFT or RIGHT)



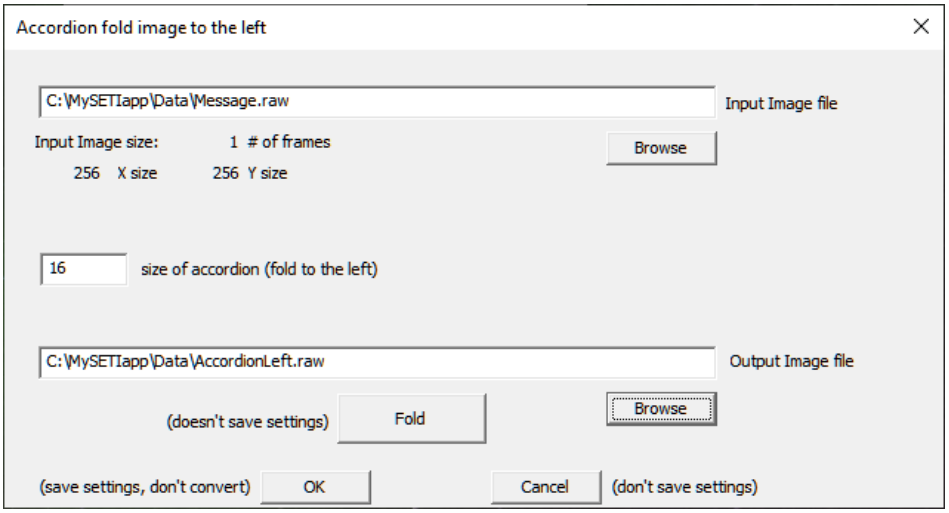
The input image is folded at the specified column either LEFT or RIGHT. The LEFT fold is shown above. If the image is not folded exactly in the center the resulting image is enlarged to accommodate the fold location. The resulting new image size is also reported. If there are multiple frames in the image file then each frame is identically folded. The pixels in the fold image are added together. If 2 pixels overlap as a result of the fold the new pixel value is the sum of the 2 overlapped pixels.

Fold along row (UP or DOWN)



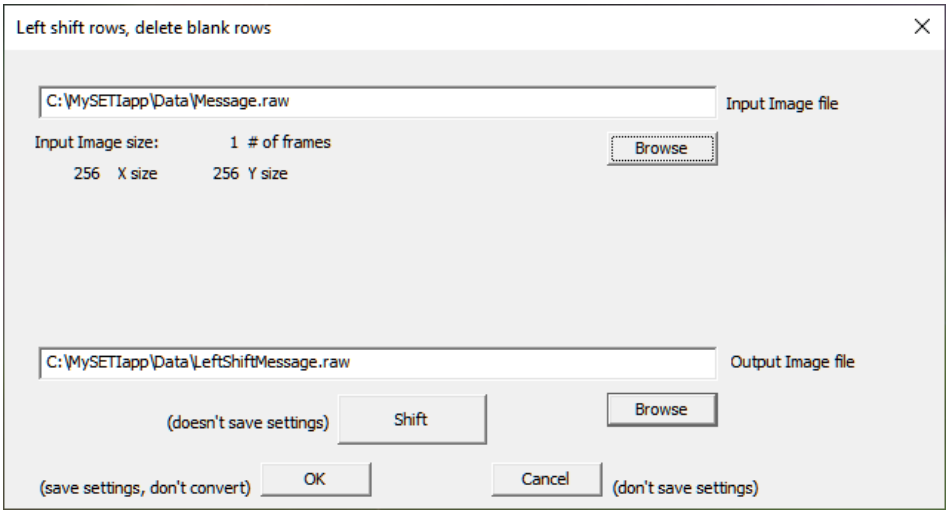
The input image is folded at the specified row either UP or DOWN. The UP fold is shown above. If the image is not folded exactly in the center the resulting image is enlarged to accommodate the fold location. The resulting new image size is also reported. If there are multiple frames in the image file then each frame is identically folded. The pixels in the fold image are added together. If 2 pixels overlap as a result of the fold the new pixel value is the sum of the 2 overlapped pixels.
Restrictions: Image must be even sized. Odd sized folds planned in future update.

Accordion fold (LEFT or RIGHT)



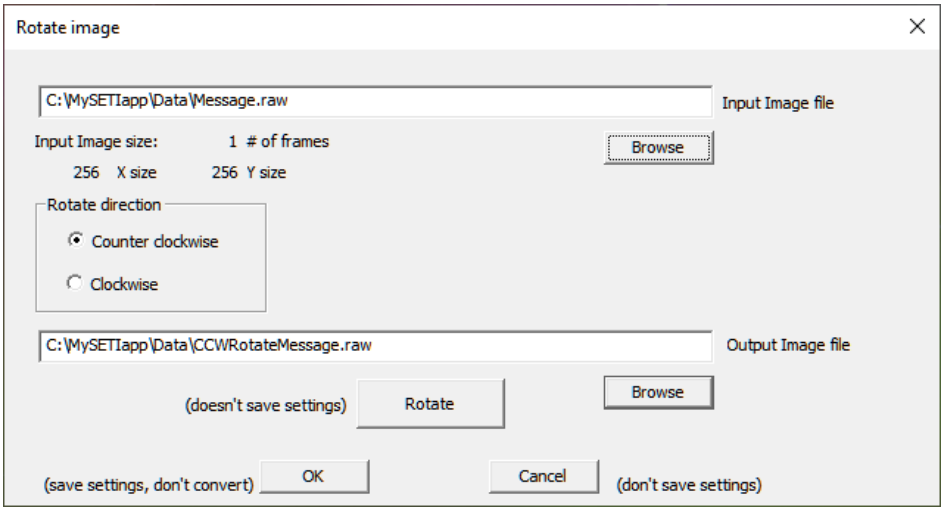
This function does an accordion fold an image along a vertical axis. It folds the left or right side of the accordion fold to the opposite side of the fold. The folded file is 1/2 the width of the unfolded image. The width of the input image must be divisible by the accordion size. The input image file width must be even. The accordion size must also be even. Think paper being cut into strips that are the width of the accordion size. Then the strip is folded and the folded strips stuck back together.
Restrictions: Image must be even sized. Odd sized folds planned in future update.

Left shift rows, delete blank rows



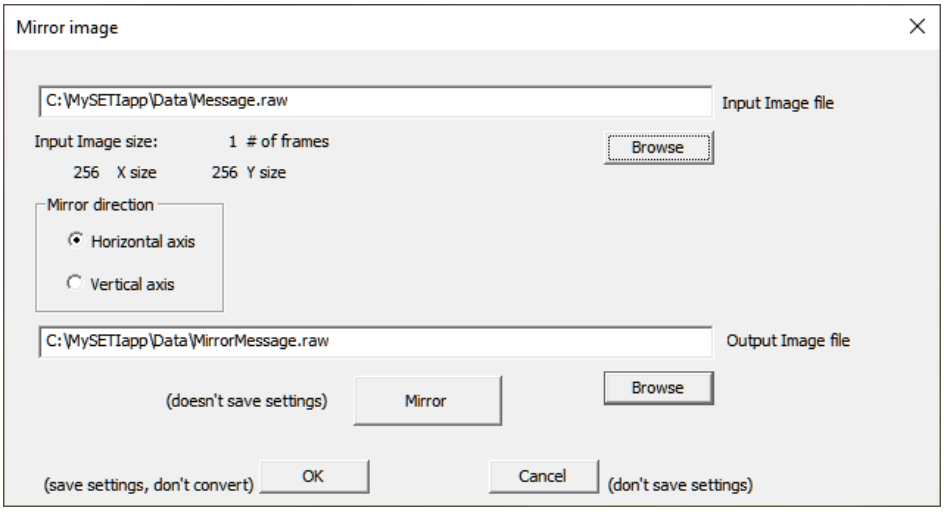
This function shifts a row to the left so that leading 0s are eliminated. Blank rows are also deleted. This operation does not appear to applied the 'A Sign In Space' message. It is useful for asynchronous serial streams with varying packet lengths.

Rotate Image



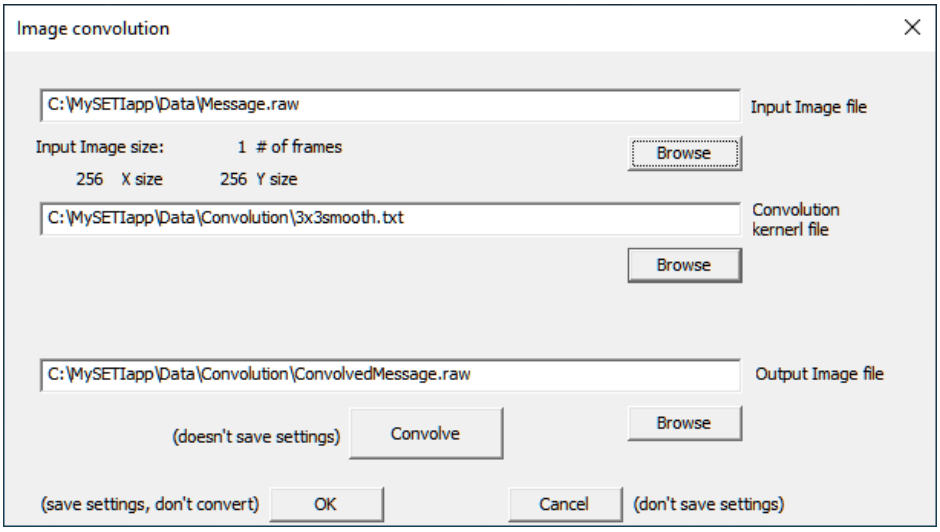
This rotates an image counter clockwise or clockwise. If the image is not square the new image X and Y size will change to reflect the rotation.

Mirror image



This mirrors an image counter on the specified axis. The If the image is not square the new image X and Y size will change to reflect the rotation.

Convolution



This applies a convolution kernel to the specified image. It does not scale the results afterwards. This function does a kernel convolution on the input image. It does not scale the results afterwards. The convolution kernel is read in from a text file. The kernel weights are floating point numbers and can be less than 0. Kernel sizes do not have to be square and can even be linear (such as a convolution kernel for a 1D image file). If you are not familiar with convolution there is material online and in image processing textbooks that explain it. The border of an image that has been convolved may be missing data due to the convolution. The size of the kernel will dictate how many rows and columns on the border of the image are affected. These will be 0 filled in the output file.

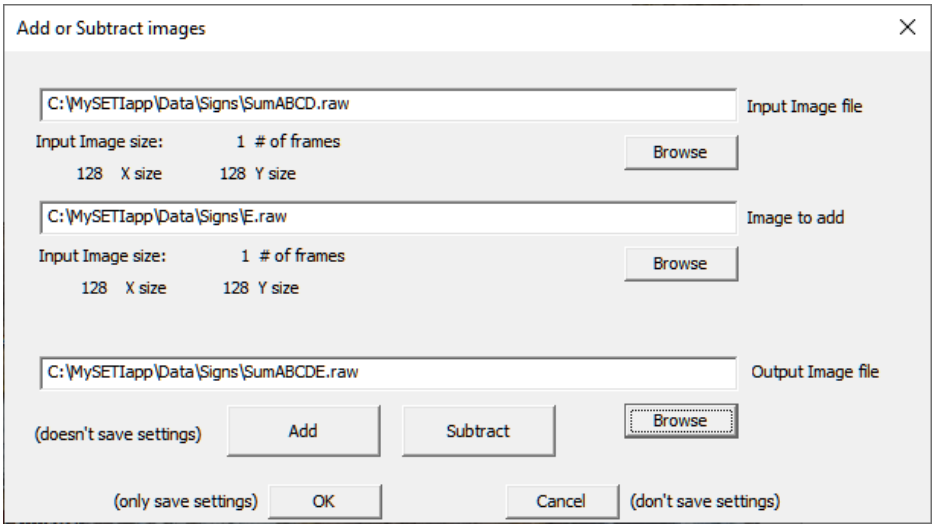
Kernel file format

n,m	Kernel size, n wide by m long
w1 w2 w3	List n*m long of weights white space delimited
optional description	

Example

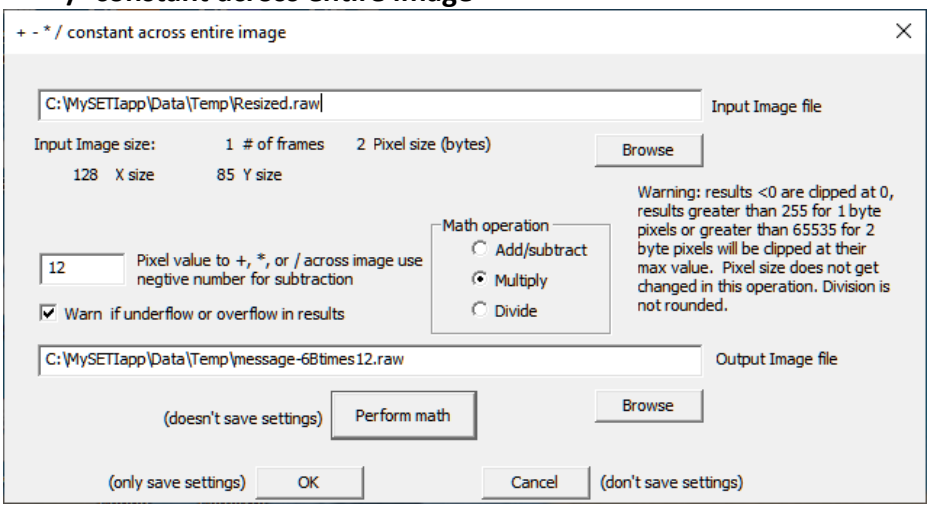
3,3
0.071428571 0.142857143 0.071428571
0.142857143 0.142857143 0.142857143
0.071428571 0.142857143 0.071428571
3x3 kernel, smoothing using weighted average

Add/Subtract images



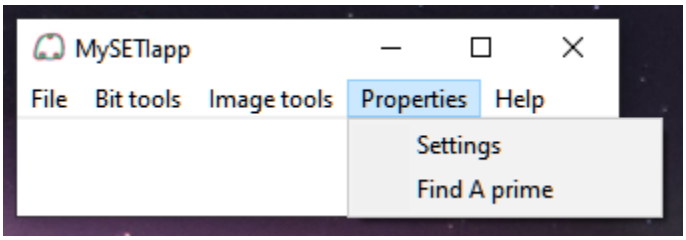
This add or subtracts the second to/from the first image. Both input files must have the same X,Y size and number of frames. If there is more than 1 frame in a file than the summation is frame by frame from each file. If the results for pixel is less than 0 it will be set to 0.

+ - * / constant across entire image

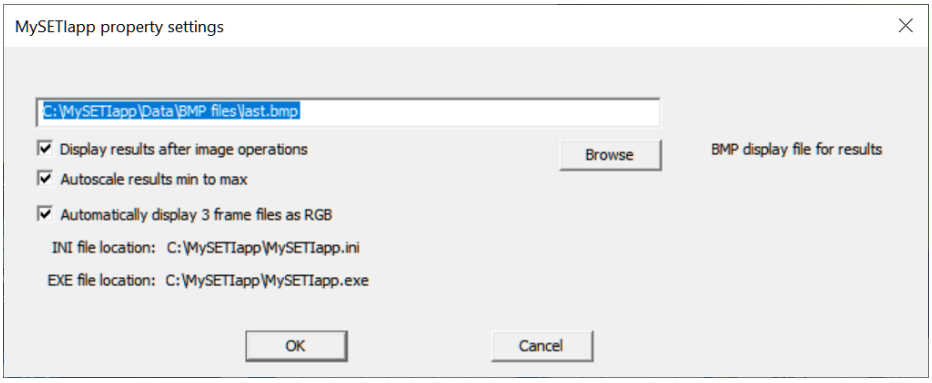


This function adds a constant value across the entire image. The value can be < 0. If the result for s pixel is less than 0 then the pixel is set to 0.

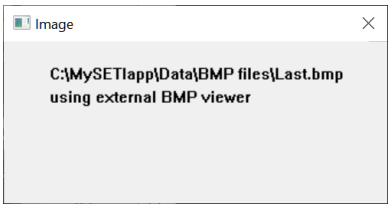
Properties and Help menu



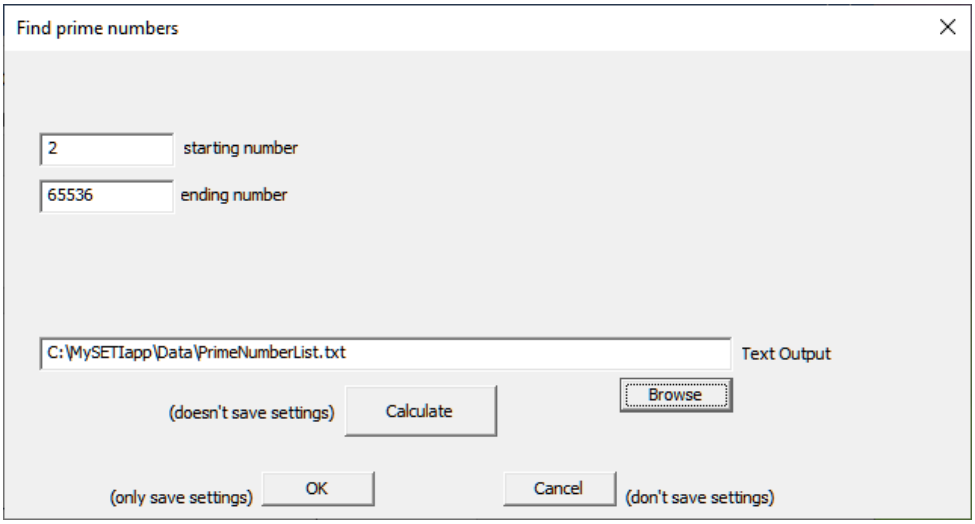
Settings



The Properties menu allows you to set program settings. Operations that generated an output image file will also generate a BMP file in the folder using the name specified in this settings dialog. The Display results after image operations must be checked for this to happen. This will also cause a bitmap image to be displayed in a separate window.

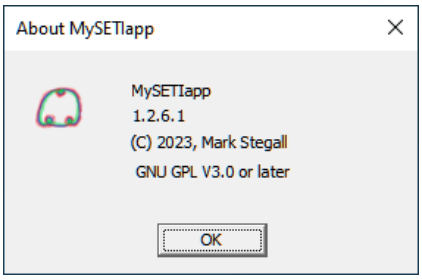


Find A prime



This generates a file containing the list of prime number in the range of numbers specified.

About



This displays the application's current About dialog.

Version information

V1.0.0.1	Aug 20,2023	Initial release
V1.1.0.1	Aug 22,2023	Added file type specifications to open/save dialogs Added resize image Added image decimation Added external BMP viewer to show result of an image operation
V1.2.0.1	Aug 31,2023	Added Convert text to packed bitstream file. Added image stats reporting not just image header stats. Corrected error handling of reorder list, when entry in kernel is out of bounds, file closure on error. Corrected error handling of reorder list, when entry in kernel is out of bounds, file closure on error. Clean up of ImageDlg to just rely on external viewer. Use of Windows default viewer for BMP display works adequately.
V1.2.1.1	2023-09-06	Added import BMP to image file (.raw). Must be 1 bit or 8 bit Added HEX text file to binary file Decimation process, corrected incorrect calculation for new Y dimension. Corrected ImageDlg to display results file only once.
V.1.2.2.1	2023-09-06	Correction, check that the path exists when specifying file is open/save dialog Correction, change default folders\filenames Correction, change default app settings, removed Autosize setting Corrected bug in V1.2.1 when importing BMP file which resulted in incorrect frame size.
V.1.2.3.1	2023-09-06	This release was skipped because of a mix-up of V1.2.1 and V1.2.2
V1.2.4.1	2023-09-09	Added, Import of CamIRa IMG files (not applicable to A Sign in Space project) Added, Add/Subtract constant from images Added, batch mode for Bit stream to Image file Changed, Sum images to Add/Subtract images Cleanup of file open/save dialog filename handling
V1.2.5.1	2023-09-10	Added standard image decimation (summation) Updated, handling of default directories Updated, Tab order and default buttons in all dialogs
V1.2.6.1	2030-09-24	Added, Increase image size by replication of pixels. Added, algorithm driven reordering, only 5 algorithms initially implemented. Added, Block symbol/phrase extraction from either 1D or 2D image file. Changed, HEX file output to include # bytes to skip at the beginning of the input file. Changed, Display Image/BMP file to include 16 and 32 bit image files using a scaled 8bpp BMP file. (This is not applicable to the 'A Sign in Space' project.) Correction, Display Image/BMP file was not displaying some BMP files.

V1.2.7.1 2023-10-01 Added missing description for the Block symbol extraction.
Added, Inversion of inputs bits in bitstream when using Bit stream to text and Bit stream to image operations.
Added, Insert/add image into an existing image, to able to recreate 'A Sign in Space' starmap from the 5 sign image or to make up one's own message.
Correction, 2d symbol extraction requires image y size to be divisible by the y symbol size. Change, Block symbol extraction now allows highlighting phrases.
Change, added skeleton for adding 5 more algorithms to algorithm reordering operation.

V1.2.8.1 2023-10-18

Added input2 image file information to dialog for Append end image operation
Added, Space protocol Packet extraction from a TM SPP stream file.
Added, Export Image to bitstream file
Added 4 new algorithms to reorder using algorithm
- Incremental row shoft with wrap aorund
- n Stripes
- Shift (rotate) Rows
- Shift (rotate) Columns

Added Filesize function to filefunctions.cpp
Changed, Append image end, no longer requires Ysize to be the same unless the frames are being added to the end of the first input image.
Changed, bit sequence report to also include 0s as a sequence.
Changed, Add/subtract constant to be add/subract/multiply/divide constant operation
Changed, Add filesize to most bit stream operation dialogs
Correction, 16 bit image pixels were being treated as signed instead of unsigned in certain circumstances.
Correction, reshource.h files to stop ID value are not duplicated. Duplication of ID numbers is something the resource editor automatically does and requires occasional cleanup to keep unwanted GUI actions and compilation errors from occurring. This included checking that a resource ID does not get assigned 65535, this is reserved for IDC_STATIC as -1 (ID #is 16 bits).
Corrected, various Dialog titles

V1.2.9.1 2023-10-31

Added, Bitstream Dialog, Remove NULL bytes sequences from bit stream file
Added, Image Dialog, Batch processing for reordering using text file with filenames
Changed, Reorder using algorithm dialog.
- Added algorithm to reorder using algorithm, Block output [MxN] reordering
- Added Invert algorithm option
- Added parameter P3 (for future use, not used in this release)

Correction, fixed bug in Extract image, when extracted image extended past vertical end of input image.

V1.2.10.1 2023-11-6

Added, dialog for Add kernel to image on a kernel based grid

Changed, Reorder using algorithm dialog.

Add algorithm, split image left/right

Changed, Global settings dialog, added auto generate .PNG file flag whenever .BMP generated.

Changed, Text to bitstream file, added output byte bit order flag

Changed, Bitstream to binary image

added input byte bit order flag

dialog changes to clarify which bit order flag applies to input and output files

Changed, all pixel reordering dialogs, added invert algorithm flag

Correction, Pixel reordering operations, fixed error so that image files with pixels size of 2 or 4 are saved correctly.

V1.2.11.1 2023-11-8

Added, Reorder blocks[MxN] in image using text kernel file

V1.2.12.1 2023-11-20

Added, Prime Number list generator

Added, Batch Extract image

Correction, Fixed extract image when extracting multi frame image files