GPR_PROC

Version: 1.03.30.00
Last revision date: 3-30-2000
Description: GPR_PROC processes digital GPR data. The input to this program is a "CMD" file, an ASCII text file containing keywords (or commands) which are discussed in a section below. There is no graphic display of the data. To display the processed data, use programs such as GPR_DISP.EXE or FIELDVEW.EXE. This program also does not convert storage formats. Output files have the same storage format as the input files. GPR_CONV.EXE can be used to convert between popular storage formats and/or user-defined formats. If you need to select a subset of traces from a file or change the number of samples per trace then use GPR_SAMP.EXE.

PROCESSING OPERATIONS

The processing operations are executed in the order they are given in the keyword file. They can be called in any order and some thought (and experimentation) should be given as to the proper sequence. They can also be called more than once for a maximum number of 100 processing operations. One way to evaluate the effect of processing on the original data is to perform one or more operations and then subtract that data set from the original one using GPR_DIFF.EXE.

- Remove range gain
- Add range gain
- Apply a low-, high-, or band-pass frequency-filter to the traces
- Remove a global background (average) trace
- Remove the global foreground traces (shows the background trace)
- Remove a sliding-window “background” (average) trace (reduces sub-horizontal features)
- Remove the sliding-window “foreground” traces (this is a "dip filter")
- Adjust the trace mean value up or down
- Shift the samples up or down
- Stack the traces (reduces the file size)
- Scale the trace amplitudes (includes sign reversal)
- Smooth horizontally over several traces
- Smooth vertically over samples within a trace
- Apply a spatial ("horizontal") median filter
- Apply a temporal ("vertical") median filter
- Transform traces to instantaneous amplitude
- Transform traces to instantaneous power
- Equalize all traces

THE KEYWORDS

Following is the list of keywords and their default values. The documentation format is:
"KEYWORD: **keyword** = default value".
Look at GPR_PROC.CMD as an example command file with correct usage and default keyword values. The file GPR_PROC.CMD has most comments stripped out, and GPR_PROC.CM_ has all comments removed.

****************************** PROGRAM CONTROL ********************************
KEYWORD: **batch** = "FALSE"
Place program in batch mode (no pauses) if "TRUE". If set to "FALSE", the program will pause after the keyword values are displayed and ask if you want to continue. After the data are processed, the program will end automatically.

***************** SPECIFICATION OF INPUT AND OUTPUT DATA ********************
Many data files can be read in, but at least one must be given. The data storage format is determined by inspecting the file. If the program cannot recognize one of the three storage formats below then that file is skipped. All GPR data are converted to 64-bit floating-point data for internal use. Then the data are converted back to the native format for output. Data must be stored in a binary format. This program cannot read GPR data stored in text files.

Recognized storage formats are:
DZT - GSSI DZT file
DT1 - Sensors & Software pulseEKKO file with a matching HD text file
SGY - SEG SEG-Y format

DT1 and HD files are assumed paired, i.e. both have the same filename with different extensions. So, if a data file with a ".DT1" extension is specified, the ".HD" filename will be assumed. Only DT1/HD files must have those filename extensions.

KEYWORD: **num_input_files** = 0
Replace the 0 with the number of files you wish to process. There is no set limit to the number of files. There must be the same number of output as input files. The first output file corresponds to the first input file.

KEYWORD: **input_filelist[]**

KEYWORD: **output_filelist[]**

Add an equal sign, =, then list the filenames after the brackets. The list can extend across multiple lines. Output filenames cannot be duplicated in the output list. Input filenames can be duplicated in the input list but cannot appear in the output list after they appear in the input list. Here are some examples; they would all be read in the same.

```
input_filelist[] = file1.dzt file2.dzt file3.dzt file4.dzt file5.dzt

input_filelist[] =
        file1.dzt file2.dzt file3.dzt file4.dzt file5.dzt

input_filelist[] = file1.dzt
        file2.dzt file3.dzt
        file4.dzt file5.dzt

input_filelist[] =
        file1.dzt
        file2.dzt
        file3.dzt
        file4.dzt
        file5.dzt
```

KEYWORD: **channel** = 1
This keyword applies to multi-channel GSSI DZT files only. Note that output files are single-channel only. This is the channel to use in multi-channel data sets. GSSI data can have up to 4 channels (channel = 1, 2, 3, or 4). PulseEKKO and RAMAC data contain only 1 channel and this keyword is ignored.

********************** SELECTING PROCESSING OPTIONS ************************
This group determines how the data are processed. The keywords appear here in approximate alphabetical order. Processing order is determined by the order these commands are found in the CMD file. You can use any order you want and repeat the values for up to a maximum of 100 processes. The default values here indicate that nothing will be done to the data. Data are converted to floating points so that dynamic range is not an issue during processing.

KEYWORD: **amp_adjust** = "INVALID_VALUE"
If set to a real value other than the default expression "INVALID_VALUE", then the average trace amplitude (that is, the mean value) is adjusted to this value. Because the data values are converted to floating point and unsigned data are first adjusted by subtracting the middle value of the data type, all data types will tend to have a mean value near zero. For example, 16-bit unsigned data have values between 0 and 65535. When converted to doubles, the value 32768 is subtracted from each data point, so the range is now –32786 to 32767. Amp_adjust forces the trace mean to be the new value according to the expression

new_value = old_value + (new_mean - old_mean).

The "old_mean" value is calculated for each trace.

############ NOTE #############
A value of 0 will force the mean of each trace to have the middle value of the data type. A value other than 0 will have the effect of decreasing or increasing all amplitudes. This feature is useful in removing "random" DC shifts in the data.
################################

KEYWORD: **amp_scale** = 1.0
This option changes the amplitude of the trace sample. If it is not equal to 0, then it is the value to multiply all samples by. Use 0 or 1 for no scaling. A –1 will reverse the "sign" of the traces.

KEYWORD: **glob_bckgrnd_rem** = "FALSE"
If set to TRUE (in double quotes like above), then a "global" background trace is removed from the data. The background trace is the average trace determined by adding all traces together and dividing by the number of traces. This is also called stacking. The stacking process enhances coherent signal and reduces randomly varying signal (or noise). In this case, the coherent signal is the horizontal banding often seen in GPR data (what we call system noise) and the randomly varying signal is the received radar signal from the subsurface. The appearance of the data is often improved by removing the horizontal banding. Caution must be used, however, with small data sets (less than about 1000 traces) or data that has strong natural horizontal reflectors. Frequency filtering may be an alternative for some data.

KEYWORD: **glob_forgrnd_rem** = "FALSE"
If set to TRUE (in double quotes like above), then a background trace is calculated like explained above but is assigned to all the traces. I call this foreground removal. This option allows you to see what is removed by the glob_bckgrnd_rem option.

KEYWORD: **high_freq_cutoff** = -1.0
Frequencies, in MHz, above this value will be removed from the data. This is called low-pass (or high-cut) filtering. It removes higher frequencies from the data. If less than 0, then there is no high-cut filtering requested. The Fourier transform (FFT) of each trace is used. A band-pass filter can be constructed by defining appropriate values for both low_freq_cutoff and high_freq_cutoff. If the value for high_freq_cutoff is greater than 0 it must also be greater than low_freq_cutoff, else no frequency filtering will occur.

############ NOTE #############
low_freq_cutoff and high_freq_cutoff must be assigned in pairs. Both must appear with equal signs and have zero or positive values or they are ignored.
################################

KEYWORD: **hsmooth** = 0
If this value is greater than 0, then it is the half-width of a Hanning window to be used for horizontal smoothing across traces. The Hanning window assigns the middle value a weight of 1.0 and the end values a weight of 0.0. Values in between are assigned a weight based on the function (0.5 - 0.5 * cos(t)). This operation tends to smooth data out horizontally. The larger the window is the smoother the data will appear. The window must be an odd value so 1 will be added to even values.

KEYWORD: **inst_amp** = "FALSE"
If this keyword is set to "TRUE", then the amplitudes of each trace are converted to instantaneous amplitude. An analytic function is constructed using the original trace as the real component and its Hilbert transform as the imaginary component. The modulus of the complex function (the square root of the sum of the squares of the real and imaginary components) is called the instantaneous amplitude of the function. For GPR data it measures the reflectivity strength, reducing the appearance of random signal in the data.

KEYWORD: **inst_pow** = "FALSE"
If this keyword is set to "TRUE", then the amplitudes of each trace are converted to instantaneous power, or energy. An analytic function is constructed using the original trace as the real component and its Hilbert transform as the imaginary component. The square of the modulus of the complex function (the square root of the sum of the squares of the real and imaginary components) is used. For GPR data it measures the total energy of the GPR signal at an instant in time. The effect on the appearance of the data is similar to converting to instantaneous amplitude, but noise is reduced even further.

############ NOTE #############
Only one, inst_amp or inst_pow, will be used. It is the first one that is found set to "TRUE".
################################

KEYWORD: **low_freq_cutoff** = -1.0
Frequencies, in MHz, below this value will be removed from the data. This is called high-pass (or low-cut) filtering. It removes lower frequencies from the data. If less than 0, then there is no low-cut filtering. The fast Fourier transform (FFT) of each trace is used. A band-pass filter can be constructed by defining appropriate values for both low_freq_cutoff and high_freq_cutoff.

############ NOTE #############
low_freq_cutoff and high_freq_cutoff must be assigned in pairs. Both must appear with equal signs and have zero or positive values or they are ignored.
################################

KEYWORD: **num_gain_off** = 0
If this keyword is set to a value greater than or equal to 2, then it is the number of breakpoints to use for gain removal. Because the strength of the radar signal attenuates rapidly, a time-varying gain is often applied to make reflections near the bottom of the trace (later arrival times) more visible. This option (and gain_off[]) removes that gain or part of it. If set to 0, then no gain is removed. For example, if there are 512 samples in a trace and there are 8 breakpoints, the 8 dB-values specified in gain_off[] will be assigned to sample numbers 0, 73, 146, 219, 292, 365, 438, and 511 (the step is 511/8). The dB values are linearly interpolated between these values. All dB values are then converted to linear values and applied to the data. Note that GPR_DISP also adds and removes gain but without affecting the stored data.

KEYWORD: **gain_off[]**
This is the set of floating point values for the gain that will be removed. NOTE: These values are in decibels, dB! For example, to multiply data by 1000, a decibel value of 60 is used (20 * log(1000) = 20 * 3). To multiply by 2 use 6; by 4 use 12; by 8 use 18. To decrease data by 10 (i.e. multiply by 0.10), a decibel value of –20 is used (20 * log(0.1) = 20 * -1). S&S DT1 files often do not have gain applied. GSSI DZT files usually do have gain applied and the values can be known by inspecting the file header with programs such as GPR_RHDR.EXE or DZT_RHDR.EXE.

############ NOTE #############
Changes to the range gain of DZT files are not reflected in the file header. The file header will still show the original gain applied to the data at record time. Processing operations and values are noted in the text/comment areas of the file header. Use program S10_EDHR.EXE to change the range gain stored in the DZT file header.
################################

REMEMBER to add the equal sign, =, if using this option.

```
Example:     num_gain_off = 2
             gain_off[] = 6 15
```

KEYWORD: **num_gain_on** = 0
If this keyword is set to a value greater than or equal to 2, then it is the number of breakpoints to use for adding gain. HERE Because the strength of the radar signal attenuates rapidly, a time-varying gain is often applied to make reflections near the bottom of the trace (later arrival times) more visible. This option removes that gain. If == 0, then no gain is added. For example, if there are 512 samples in a trace and there are 2 breakpoints, the 2 dB values specified in gain_on[] will be assigned to sample numbers 0 and 511. The dB values are linearly interpolated between these values. All dB values are then converted to linear values and applied to the data. REMEMBER that GPR_DISP also adds and removes gain without affecting the stored data.

KEYWORD: **gain_on[]**
This is the set of floating point values for the gain, which will be added. NOTE: These values are in decibels, dB! For example, to multiply data by 50, a decibel value of 34 is used (20 * log(50)= 20 * 1.7). To decrease data by 20 (i.e. multiply by 0.05), a decibel value of –26 is used (20 * log(0.05) = 20 * -1.3). S&S DT1 files often do not have gain applied. GSSI DZT files usually do have gain applied and the values can be known by inspecting the file header with programs such as GPR_RHDR.EXE or DZT_RHDR.EXE.

REMEMBER to add the equal sign, =, if using this option.

```
Example:     num_gain_on = 2
             gain_on[] = 6 15
```

KEYWORD: **preprocFFT** = "TRUE"
If set to "TRUE" and filtering is to be done (low_freq_cutoff and high_freq_cutoff are greater than or equal to 0), then the start and end of each trace is reduced to the median value using a Hanning cosine taper. Normally this value should be set to "TRUE".

KEYWORD: **samp_slide** = 0
This is the number of locations to slide sample values down (later in time, a positive value) or up (earlier in time, a negative value) for all traces. Sample sliding does not change the sample rate or the number of samples per trace. The median value of the trace data type (for example, 32768 for unsigned 16-bit data) is assigned for "new" samples on the top or bottom. Sample values that "slide" off the trace are lost. All traces are "slid" the same. The "top" sample in a trace is the first sample (earliest in time).

############ NOTE #############
The sample sliding operation DOES NOT change the range gain records in DZT headers (or comment area of other storage types). To keep the DZT header information correct in the text/comment area, first remove range gain from DZT files using num_gain_off and gain_off[] (see above), slide the samples, then add gain back to data (it can be the same or different) using num_gain_on and gain_on[] (see below). To calculate new dB values for the "slid" samples, linear interpolation can be used between the old gain dB values. Use program S10_EDHR.EXE to change the range gain stored in the DZT file header.
################################

KEYWORD: **spatial_median** = 0
If this value is greater than 0, then it is the width of a median filter to be applied "horizontally" across the traces. A median filter assigns the middle value of the set of values found in the window to the central data point. This is useful in removing "spike" values (or rapid changes) that may occur from trace to trace.

KEYWORD: **stack** = 0
This option reduces the number of output traces. If greater than 0, then it is the number of traces to stack into an average trace. The average trace replaces the group of traces. NOTE: grid stacking ignores trace headers and passes the values from the first stack trace through.

############ NOTE #############
No attempt is made to synchronize trace header values with the final stacked trace. This means that GSSI markers may be lost and some info in S&S and SEG-Y trace headers may be wrong. To preserve marker information, use the program GPR_STAK.EXE.
################################

KEYWORD: **temporal_median** = 0
If this value is greater than 0, then width of a median filter to be applied "vertically" down each traces. A median filter assigns the middle value of the set of values found in the window to the central data point. This is useful in removing "spike" values (or rapid changes) that may occur from sample to sample in a trace due to external noise.

KEYWORD: **trace_equalize** = -1
This is a flag indicating how to equalize trace amplitudes. If equalization is selected then the sum of the absolute values of all samples in a trace is made the same for all traces. So if the trace that is selected to be used for equalization has a sum of all absolute values of 500000, then the sum of absolute values of every trace in the data set will be set to this using a multiplying factor. The valid values for this keyword are:

-1 = no equalization (default)
 0 = use first trace

-2 = use middle trace
-3 = use last trace
n = use trace n

KEYWORD: **vsmooth** = 0
If this value is greater than 0, then it is the half-width of a Hanning window to be used for vertical smoothing along samples in each trace. The Hanning window assigns the middle value a weight of 1.0 and the end values a weight of 0.0. Values in between are assigned a weight based on the function (0.5 - 0.5 * cos(t)). This operation tends to smooth the samples out in a trace. The larger the window is the smoother the data will appear. The window must be an odd value so 1 will be added to even values.

KEYWORD: **wind_bckgrnd_rem** = 0
If this value is greater than 1, then it is the size of sliding window to be used for background trace removal calculations, as opposed to using the entire data set to calculate a background, or average, trace. This operation will remove small horizontal features (coherent signal) from the data. This feature can be used to expose reflections that dip at high angles, REMOVING most reflections due to hydrogeologic sources. The appropriate size of the window is unique for each data set. The window must be an odd value so 1 will be added to even values.

KEYWORD: **wind_forgrnd_rem** = 0
If this value is greater than 1, then it is the size of sliding window to be used for background trace calculations, as opposed to using the entire data set to calculate a background, or average, trace. Unlike wind_bckgrnd_rem, this operation removes the foreground, that is, the background trace is assigned to the center trace in the sliding window (rather than being subtracted from the data). This operation will remove high-angle reflections (a dip filter) to expose the sub-horizontal hydrogeologic features more clearly. The appropriate size of the window is unique for each data set. The window must be an odd value so 1 will be added to even values.

Usage: GPR_PROC cmd_filename
Required command line arguments:
cmd_filename - The name of the keyword file.
Optional command line arguments (do not include brackets): none
Examples:

```
gpr_proc pfile1.cmd
```