HUNT ENGINEERING
Chestnut Court, Burton Row,
Brent Knoll, Somerset, TA9 4BP, UK
Tel: (+44) (0)1278 760188,
Fax: (+44) (0)1278 760199,
Email: sales@hunteng.co.uk
http://www.hunteng.co.uk
http://www.hunt-dsp.com

HUNT ENGINEERING

HEL_UNPACK.LIB

USER MANUAL

Unpacking / data access library for Hunt Engineering ADC & DAC modules.

Many multichannel devices such as ADCs multiplex channels of data onto a single HERON FIFO. These channels need to be separated before processing, or packed together before output. The library described here provides efficient assembly-coded routines to pack or unpack the data to/from separate arrays, and optionally to convert it to a different format.

Software Rev 1.4
Document Rev 1.4
R.Weir 14/11/01

COPYRIGHT

WARRANTIES LIABILITY and INDEMNITIES

HUNT ENGINEERING warrants the hardware to be free from defects in the material and workmanship for 12 months from the date of purchase. Product returned under the terms of the warranty must be returned carriage paid to the main offices of HUNT ENGINEERING situated at BRENT KNOLL Somerset UK, the product will be repaired or replaced at the discretion of HUNT ENGINEERING.

Exclusions - If HUNT ENGINEERING decides that there is any evidence of electrical or mechanical abuse to the hardware, then the customer shall have no recourse to HUNT ENGINEERING or its agents. In such circumstances HUNT ENGINEERING may at its discretion offer to repair the hardware and charge for that repair.

Limitations of Liability - HUNT ENGINEERING makes no warranty as to the fitness of the product for any particular purpose. In no event shall HUNT ENGINEERING'S liability related to the product exceed the purchase fee actually paid by you for the product. Neither HUNT ENGINEERING nor its suppliers shall in any event be liable for any indirect, consequential or financial damages caused by the delivery, use or performance of this product.

Because some states do not allow the exclusion or limitation of incidental or consequential damages or limitation on how long an implied warranty lasts, the above limitations may not apply to you.

TECHNICAL SUPPORT

Technical support for HUNT ENGINEERING products should first be obtained from the comprehensive Support section www.hunteng.co.uk/support/index.htm on the HUNT ENGINEERING web site. This includes FAQs, latest product, software and documentation updates etc. Or contact your local supplier - if you are unsure of details please refer to www.hunteng.co.uk for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing support@hunteng.demon.co.uk, calling the direct support telephone number +44 (0)1278 760775, or by calling the general number +44 (0)1278 760188 and choosing the technical support option.

TABLE OF CONTENTS

V1.0 Released 22nd October 2000

1st release of library – contained unpackers for 12 & 16-bit ADCs.

V1.1 Released 8th December 2000

2nd release offered a more consistent interface with increased flexibility, extended functionality, and an improved manual. Major additions were:

- Packers for 12/14 & 16-bit DACs were introduced. These should support all ADCs up to 16-bit for the foreseeable future.
- Calling conventions for unpackers were changed to allow the wordlength-limiting mask to be passed as a parameter.

Changing the calling convention means that source-code modifications are needed to upgrade to this version of the library. However, they extend the library's abilities to cater for additional wordlength ADCs that we may supply in the future.

V1.2 Released 12th Feb 2001

This release fixed a problem when using masks that have the top but set. All masks are now defined as unsigned to prevent this problem.

V1.3 Released 20th April 2001

Added HEL_UnpackSe16BitYchan functions, to support HERON -IO. The functions unpack and sign-extend a 12-bit data value.

Added HEL_PackBin2h16BitYchan functions, to support GD14 data formats. It accepts an array of signed shorts, and packs them into proper GD14 data format.

V1.4 Released 14th November 2001

HEL_UnpackSe16BitYchan functions now support any data length, not only 12-bit data. The functions unpack and sign-extend any xx-bit data value, with xx in [15..1].

In many cases, I/O modules used with HERON systems pack their data into a single stream. This allows multi-channel modules to operate with a single FIFO. It is the only way that a GDIO module can implement multiple channels – the GDIO specification includes only one channel; and it is a very practical way of ensuring data remains synchronised through the system.

However, this data format is not always ideal for processing. The incoming data stream may include tag or address information, and the multiplex within the stream may make it difficult to access consecutive points with signal processing algorithms. It is usually most appropriate to unpack the data into separate arrays.

For output devices, the data may include control information – such as the last channel bit, or channel address. This must be added before the data is multiplexed for output.

This library implements a suite of routines which implement unpacking / packing for the data types used in Hunt Engineering I/O modules. In addition, the data's format may be converted to 2's complement if required. This is performed as it is unpacked, with almost no time penalty.

Where do I find the library?

When you have made a normal software installation from the HUNT ENGINEERING CD, the library and header file for HEL_Unpack have been placed in the heron_unpack directory under your API installation directory. i.e. %HEAPI%\heron_unpack

The "Create New HERON Project" plug-in for Code Composer Studio will automatically include this library into a new project, and will add the directory to the include path. This means that the functions in this library can be used in any project without concern.

ADC Unpackers

Data Format Conversion

There are two sets of unpackers, covering 16-bit and 32-bit data sources. The 32-bit unpackers assume that a single sample is contained in one half of a 32-bit word, as in the case with the 16-bit ADCs; while the 16-bit unpackers assume that the sample is contained within some a 16-bit word. In either case, the library can extract different wordlengths – for example, the 16-bit unpackers can extract 12 or 14-bit samples.

The unpackers are split into two groups. The first group unpack the data directly, and perform no format conversion. The second group convert the data from "offset binary" to "2's complement". Offset Binary is a common format used on many ADCs & DACs.

Data Buffer Formats

Output data is in consecutive linear addressed buffers. It is assumed that the output buffers are consecutive – see example.

The output data is 16-bit. Buffers allocated must be large enough to receive the results.

Assumptions & Notes

1. All code is interruptible to ensure maximum performance in HERON systems.
2. It is assumed that the first sample in the input buffer is from Channel #0, and that all subsequent samples are from consecutive channels. This is the normal startup sequence for HERON IO systems.
3. The unpacker functions will not detect a FIFO overflow. If FIFO overflow occurs, synchronisation will be lost and must be regained externally.
4. In all cases it is assumed that each input buffer will contain at least 8 samples, and that each output buffer will contain at least 2 samples.
5. It is also a condition that all buffers should contain an even number of samples and be aligned to 32-bit boundaries. This allows the library to treat data as 32-bit values, allowing faster data transfers.

Alignment of arrays

Many of the functions use arrays of shorts that must be aligned to 32-bit boundaries. This is mentioned in the tables below along with **. This means that the data must be forced to be aligned as follows:

```
/* allocate the errors array */
#pragma DATA_ALIGN(errors,4);
short errors[NO_OF_CHANNELS];
```

HEL_Unpack16bitYchan

Syntax:

```
#include        "HEL_Unpack.h"
…
```

HEL_Unpack16bit1chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack16bit2chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack16bit3chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack16bit4chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack16bit8chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack16bit12chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack16bit16chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);

Description:

This group of unpackers take a buffer of data from a 12-bit source, typically an ADC, and split it into multiple output buffers – one per channel.

The data source must be compatible with Hunt Engineering 12-bit ADCs (right justified, packed in 16-bit words; top 4 bits used as status).

The top four bits are cleared as the data is unpacked. This allows the data to be used directly. All status information for an output buffer is logically ORed together and saved to the *errors pointer, allowing checks to be made for flags being set or overflow.

Parameters:

Input	pointer to 16-bit input buffer, containing interleaved samples from all active channels. This buffer must be aligned to a 32-bit boundary.
Bitmask	A 16-bit mask used to limit the data. For 16-bits use 0xffff; 14-bits use 0x3fff; 12 bits use 0x0fff
Output	pointer to first output buffer. Subsequent output buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.
Errors	Pointer to an array of 16 bit error values, one array entry per channel. This array must be aligned to a 32 bit boundary**. Some I/O modules include additional info in flags along with the sample data. This is ORed together for each channel and stored here.

HEL_Unpack32bitYchan

Syntax:

```
#include        "HEL_Unpack.h"
…
HEL_Unpack32bit1chan    (int *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack32bit2chan    (int *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack32bit4chan    (int *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_Unpack32bit8chan    (int *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
```

Description:

This group of unpackers take a buffer of data from a 16-bit source, typically an ADC, and split it into multiple output buffers – one per channel.

The data source must be compatible with Hunt Engineering 16-bit ADCs (right justified, packed in 32-bit words; top 16 bits used as channel markers & status).

Only the sample data is saved to the output array. Thus, the output array is half the size (in bytes) of the input array.

Parameters:

Input	pointer to 32-bit input buffer, containing interleaved samples from all active channels. This buffer must be aligned to a 32-bit boundary.
Bitmask	A 16-bit mask used to limit the data. For 16-bits use 0xffff; 14-bits use 0x3fff; 12 bits use 0x0fff
Output	pointer to first output buffer. Subsequent output buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.
Errors	Pointer to an array of 16 bit error values, one array entry per channel. This array must be aligned to a 32 bit boundary**. Some I/O modules include additional info in flags along with the sample data. This is ORed together for each channel and stored here.

HEL_UnpackBin16bitYchan

Syntax:

```
#include        "HEL_Unpack.h"
…
```

HEL_UnpackBin16bit1chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin16bit2chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin16bit3chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin16bit4chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin16bit8chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin16bit12chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin16bit16chan	(short *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);

Description:

This group of unpackers take a buffer of data from a 12-bit source, typically an ADC, and split it into multiple output buffers – one per channel. The data is converted from offset binary to 2's complement as it is unpacked.

The data source must be compatible with Hunt Engineering 12-bit ADCs (right justified, packed in 16-bit words; top 4 bits used as status).

The top four bits are cleared as the data is unpacked. This allows the data to be used directly. All status information for an output buffer is logically ORed together and saved to the *errors pointer, allowing checks to be made for flags being set or overflow.

Parameters:

Input	pointer to 16-bit input buffer, containing interleaved samples from all active channels. This buffer must be aligned to a 32-bit boundary.
Bitmask	A 16-bit mask used to limit the data. For 16-bits use 0xffff; 14-bits use 0x3fff; 12 bits use 0x0fff
Output	pointer to first output buffer. Subsequent output buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.
Errors	Pointer to an array of 16 bit error values, one array entry per channel. This array must be aligned to a 32 bit boundary**. Some I/O modules include additional info in flags along with the sample data. This is ORed together for each channel and stored here.

HEL_UnpackBin32bitYchan

Syntax:

```
#include          "HEL_Unpack.h"
…
```

HEL_UnpackBin32bit1chan	(int *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin32bit2chan	(int *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin32bit4chan	(int *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);
HEL_UnpackBin32bit8chan	(int *input, unsigned short bitmask, short *output, int samples_per_channel, short *errors);

Description:

This group of unpackers take a buffer of data from a 16-bit source, typically an ADC, and split it into multiple output buffers – one per channel. The data is converted from offset binary to 2's complement as it is unpacked.

The data source must be compatible with Hunt Engineering 16-bit ADCs (right justified, packed in 32-bit words; top 16 bits used as channel markers & status).

Only the sample data is saved to the output array. Thus, the output array is half the size (in bytes) of the input array.

Parameters:

Input	pointer to 32-bit input buffer, containing interleaved samples from all active channels. This buffer must be aligned to a 32-bit boundary.
Bitmask	A 16-bit mask used to limit the data. For 16-bits use 0xffff; 14-bits use 0x3fff; 12 bits use 0x0fff
Output	pointer to first output buffer. Subsequent output buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.
Errors	Pointer to an array of 16 bit error values, one array entry per channel. This array must be aligned to a 32 bit boundary**. Some I/O modules include additional info in flags along with the sample data. This is ORed together for each channel and stored here.

HEL_UnpackSe16bitYchan

Syntax:

```
#include        "HEL_Unpack.h"
…
 HEL_UnpackSe16bit1chan      (short *input, unsigned short bitmask, short *output,
                             int samples_per_channel, short *errors);
 HEL_UnpackSe16bit2chan      (short *input, unsigned short bitmask, short *output,
                             int samples_per_channel, short *errors);
```

Description:

This group of unpackers take a buffer of data from a xx-bit source, typically a HERON-IO, and split it into multiple output buffers – one per channel. The 'xx-' stands for a number of 1 to 15. E.g. the functions handle both 12 and 14 bit data.

The data source must be compatible with Hunt Engineering xx-bit ADCs (right justified, packed in 16-bit words, top (16-xx) bits used as status).

The top bits are cleared, as the data is unpacked. The data is then sign-extended. This means that an array of xx-bit 'raw' data samples is ready to be used as an array of signed short integers. Example: a 'raw' data value of 0xffe will be sign-extended to become 0xfffe, a short integer value –2. A data value of 0x2 will not change and stay the same. All status information for an output buffer is logically ORed together and saved to the *errors pointer, allowing checks to be made for flags being set or overflow.

Parameters:

Input	Pointer to 16-bit input buffer, containing interleaved samples from all active channels. This buffer must be aligned to a 32-bit boundary.
Bitmask	A 16-bit mask used to limit the data. For 16-bits use 0xffff; 14-bits use 0x3fff; 12 bits use 0x0fff
Output	Pointer to first output buffer. Subsequent output buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.
Errors	Pointer to an array of 16 bit error values, one array entry per channel. This array must be aligned to a 32 bit boundary**. Some I/O modules include additional info in flags along with the sample data. This is ORed together for each channel and stored here.

DAC Packers

Data Format Conversion

There are two sets of packers, covering 16-bit and 32-bit data sinks. The 32-bit packers assume that a single sample is contained in one half of a 32-bit word, as in the case with the 16-bit DACs; while the 16-bit packers assume that the sample is contained within a 16-bit word. In either case, the library can pack different wordlengths – for example, the 16-bit packers can be used with 12 or 14-bit DACs.

The packers are split into two groups. The first group pack the data directly, performing no format conversion. The second group converts the data to "offset binary" from "2's complement". Offset Binary is a common format used on many ADCs & DACs.

Data Buffer Formats

Input data is in linear addressed buffers of 16-bit samples (short data). It is assumed that the input buffers are consecutive – see example.

The output buffer must be large enough to receive the results – note that for some packers, the output data is 32-bit – hence the output buffer will be twice the size of the input buffer.

Assumptions & Notes

1. All code is interruptible to ensure maximum performance in HERON systems.
2. In all cases it is assumed that each input buffer will contain at least 2 samples, and that each output buffer will contain at least 8 samples.
3. It is also a condition that all buffers should contain an even number of samples and be aligned to 32-bit boundaries. This allows the library to treat data as 32-bit values, allowing faster data transfers.

HEL_Pack16bitYchan

Syntax:

```
#include "HEL_Unpack.h"
…
```

HEL_Pack16bit1chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit2chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit3chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit4chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit5chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit6chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit7chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit8chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);

Description:

This group of packers take multiple buffers of short data and combine to a single output buffer.

The output buffer will be compatible with Hunt Engineering 14-bit or 12-bit DACs (right justified, packed in 16-bit words; top bits used as control).

Data is limited by ANDing with bitmask before use. This can be used to ensure that short data is in-range for a 12-bit or 14-bit DAC. Data is ORed with the masks[] array before output. This array contains a mask per input buffer, so can be used to set control bits or channel address bits.

Parameters:

Input	Pointer to first input buffer. Subsequent input buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary.
Masks[]	Array of 16-bit OR masks, one per channel. Used to set control bits.
Bitmask	16-bit mask ANDed with data before output – use 0x3fff for 14-bit DACs, 0x0fff for 12-bit DACs.
Output	Pointer to 16-bit output buffer. This buffer must be aligned to a 32-bit boundary. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.

HEL_Pack32bitYchan

Syntax:

```
#include "HEL_Unpack.h"
…
```

HEL_Pack16bit1chan	(short *input, unsigned int *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit2chan	(short *input, unsigned int *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit3chan	(short *input, unsigned int *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_Pack16bit4chan	(short *input, unsigned int *masks, unsigned short bitmask, short *output, int samples_per_channel);

Description:

This group of packers take multiple buffers of short data and combine to a single 32-bit output buffer for 16-bit DACs.

The output buffer will be compatible with Hunt Engineering 16-bit DACs (right justified, packed in 32-bit words; top bits used as control).

Data is limited by ANDing with bitmask before use. This can be used to ensure that short data is in-range for a 16-bit DAC. Data is ORed with the masks[] array before output. This array contains a mask per input buffer, so can be used to set control bits or channel address bits.

Parameters:

Input	Pointer to first input buffer. Subsequent input buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary.
Masks[]	Array of 32-bit OR masks, one per channel. Used to set control bits.
Bitmask	16-bit mask ANDed with data before output – use 0x3fff for 14-bit DACs, 0x0fff for 12-bit DACs.
Output	Pointer to 32-bit output buffer. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.

HEL_PackBin16bitYchan

Syntax:

```
#include "HEL_Unpack.h"
…
```

HEL_PackBin16bit1chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit2chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit3chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit4chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit5chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit6chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit7chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit8chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);

Description:

This group of packers take multiple buffers of short data and combine to a single output buffer, converting to offset binary in the process.

The output buffer will be compatible with Hunt Engineering 14-bit or 12-bit DACs (right justified, packed in 16-bit words; top bits used as control).

Data is limited by ANDing with bitmask before use. This can be used to ensure that short data is in-range for a 12-bit or 14-bit DAC. Data is ORed with the masks[] array before output. This array contains a mask per input buffer, so can be used to set control bits or channel address bits.

Parameters:

Input	Pointer to first input buffer. Subsequent input buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary.
Masks[]	Array of 16-bit OR masks, one per channel. Used to set control bits.
Bitmask	16-bit mask ANDed with data before output – use 0x3fff for 14-bit DACs, 0x0fff for 12-bit DACs.
Output	Pointer to 16-bit output buffer. This buffer must be aligned to a 32-bit boundary. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.

HEL_PackBin32bitYchan

Syntax:

```
#include "HEL_Unpack.h"
…
```

HEL_PackBin16bit1chan	(short *input, unsigned int *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit2chan	(short *input, unsigned int *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit3chan	(short *input, unsigned int *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin16bit4chan	(short *input, unsigned int *masks, unsigned short bitmask, short *output, int samples_per_channel);

Description:

This group of packers take multiple buffers of short data and combine to a single 32-bit output buffer for 16-bit DACs. It also converts the data to offset binary.

The output buffer will be compatible with Hunt Engineering 16-bit DACs (right justified, packed in 32-bit words; top bits used as control).

Data is limited by ANDing with bitmask before use. This can be used to ensure that short data is in-range for a 16-bit DAC. Data is ORed with the masks[] array before output. This array contains a mask per input buffer, so can be used to set control bits or channel address bits.

Parameters:

Input	Pointer to first input buffer. Subsequent input buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary.
Masks[]	Array of 32-bit OR masks, one per channel. Used to set control bits.
Bitmask	16-bit mask ANDed with data before output – use 0x3fff for 14-bit DACs, 0x0fff for 12-bit DACs.
Output	Pointer to 32-bit output buffer. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.

HEL_PackBin2h16bitYchan

Syntax:

```
#include "HEL_Unpack.h"
…
```

HEL_PackBin2h16bit1chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin2h16bit2chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin2h16bit3chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin2h16bit4chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin2h16bit5chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin2h16bit6chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin2h16bit7chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);
HEL_PackBin2h16bit8chan	(short *input, unsigned short *masks, unsigned short bitmask, short *output, int samples_per_channel);

Description:

This group of packers take multiple buffers of short data and combine to a single output buffer.

The output buffer will be compatible with Hunt Engineering the 14-bit GD14 data format (right justified, packed in 16-bit words; top bit used as control).

Data is limited by ANDing with bitmask before use. This can be used to ensure that short data is in-range for 14-bit GD14 compatible data. Data is ORed with the masks[] array before output. This array contains a mask per input buffer, so can be used to set the GD14 data's control bit.

Parameters:

Input	Pointer to first input buffer. Subsequent input buffers are located immediately after this one. This buffer must be aligned to a 32-bit boundary.
Masks[]	Array of 16-bit OR masks, one per channel. Used to set control bits.
Bitmask	16-bit mask ANDed with data before output – use 0x3fff for the 14-bit GD14.
Output	Pointer to 16-bit output buffer. This buffer must be aligned to a 32-bit boundary. Do not use the input buffer as the output as this will cause data loss.
Samples_per_channel	Number of samples per channel in the input buffer.

Each of the packers / unpackers is implemented in optimized linear assembly code. Best results will be achieved using the unpackers on data in on-chip RAM, ideally in different memory blocks.

It is impossible to code a packer/unpacker so that it does not cause memory bank contention. We have tried to reduce this as far as possible, but it is the most significant factor influencing performance. Wherever possible, arrange for the input and output buffers to be in separate memory banks.

The following figurs were generated using a TMS320C6201, with data in THE SAME BANK of on-chip memory. These should therefore be pessimistic numbers. Note N is the number of samples per channel.

HEL_Unpack16bitYchan Test

1 Channel:	36 + N * 1.0
2 Channel:	52 + N * 0.8
3 Channel:	40 + N * 1.0
4 Channel:	52 + N * 0.9
8 Channel:	76 + N * 0.8
12 Channel:	72 + N * 1.3
16 Channel:	76 + N * 1.3

HEL_UnPack32bitYchan Test

1 Channel:	28 + N * 1.0
2 Channel:	24 + N * 1.0
4 Channel:	52 + N * 1.0
8 Channel:	52 + N * 1.0

HEL_UnpackBin16bitYchan Test

1 Channel:	44 + N * 0.8
2 Channel:	44 + N * 0.9
3 Channel:	44 + N * 1.0
4 Channel:	52 + N * 0.8
8 Channel:	72 + N * 1.0
12 Channel:	68 + N * 1.2
16 Channel:	76 + N * 1.2

HEL_UnPackBin32bitYchan Test

4 Channel:	20 + N * 1.0
8 Channel:	56 + N * 1.0

HEL_Pack16bitYchan Test

1 Channel:	44 + N * 0.6
2 Channel:	52 + N * 0.9
3 Channel:	56 + N * 1.0
4 Channel:	68 + N * 0.8
5 Channel:	72 + N * 0.8
6 Channel:	88 + N * 0.7
7 Channel:	104 + N * 0.8
8 Channel:	112 + N * 0.8

HEL_Pack32bitYchan Test

1 Channel:	44 + N * 1.1
2 Channel:	40 + N * 1.0
3 Channel:	48 + N * 1.3
4 Channel:	60 + N * 1.4

HEL_PackBin16bitYchan Test

1 Channel:	60 + N * 0.5
2 Channel:	72 + N * 0.8
3 Channel:	68 + N * 1.1
4 Channel:	24 + N * 1.3
5 Channel:	84 + N * 0.8
6 Channel:	88 + N * 0.8
7 Channel:	108 + N * 0.8
8 Channel:	120 + N * 0.8

HEL_PackBin32bitYchan Test

1 Channel:	60 + N * 1.3
2 Channel:	60 + N * 1.0
3 Channel:	52 + N * 1.3
4 Channel:	140 + N * 1.3

HEL_UnpackSe16bitYchan Test

1 Channel:	40 + N * 1.1
2 Channel:	56 + N * 1.1

The source of the library functions is provided on the HUNT ENGINEERING CD in the src directory. Simply follow "Getting Started" from the CD menu and select "libraries".

This source can be copied onto your hard drive, and modified or re-built as you wish.

The library is implemented as a set of linear assembly files. To build it, run the batch file "`Build_Obj_Lib.bat`".

This batch file sets the assembler options and builds the library. Note that the assembly process will give some warnings relating to large constants – these are normal and not a concern.

The Supplied Library is built for "Small" memory model. This is suitable for most applications. Please refer to the HUNT ENGINEERING technical note "Choosing a memory model" for more details. If it is necessary to have a different memory model version of the library it can be built using the sources supplied.

Technical support for HUNT ENGINEERING products should first be obtained from the comprehensive Support section www.hunteng.co.uk/support/index.htm on the HUNT ENGINEERING web site. This includes FAQs, latest product, software and documentation updates etc. Or contact your local supplier - if you are unsure of details please refer to www.hunteng.co.uk for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing support@hunteng.demon.co.uk, calling the direct support telephone number +44 (0)1278 760775, or by calling the general number +44 (0)1278 760188 and choosing the technical support option.

If you are in North America, South America or Canada, contact our strategic partner Traquair Data Systems at www.traquair.com/company/support.html for support information and contact details.