CRUNCH Library

Programmer's Manual

DataCell Limited

v4.0.1

Disclaimer

While every precaution has been taken in the preparation of this manual, DataCell Ltd assumes no responsibility for errors or omissions. DataCell Ltd reserves the right to change the specification of the product described within this manual and the manual itself at any time without notice and without obligation of DataCell Ltd to notify any person of such revisions or changes.

Copyright Notice

Copyright ©1994-1999 DataCell Ltd and Active Silicon Ltd. All rights reserved. This document may not in whole or in part, be reproduced, transmitted, transcribed, stored in any electronic medium or machine readable form, or translated into any language or computer language without the prior written consent of DataCell Ltd.

Trademarks

"Apple", "Macintosh" and "MacOS" are trademarks of Apple Computer Inc. "AMCC" is a registered trademark of Applied Micro Circuits Corporation. "Dallas" is a registered trademark of Dallas Semiconductor Corporation. "Dell" is a registered trademark of Dell Computer Corporation. "Flash Graphics" and "X-32VM" are trademarks of Flashtek Limited. "IBM", "PC/AT", "PowerPC" and "VGA" are registered trademarks of International Business Machine Corporation. "MetroWerks" and "CodeWarrior" are registered trademarks of MetroWerks Inc. "Microsoft", "CodeView", "MS" and "MS-DOS", "Windows", "Windows NT", "Windows 95", "Windows 98", "Win32", "Visual C++" are trademarks or registered trademarks of Microsoft Corporation. "National Semiconductor" is a registered trademark of National Semiconductor Corporation. "Sun", "Ultra AX" and "Solaris" are registered trademarks of Sun Microsystems Inc. All "SPARC" trademarks are trademarks or registered trademarks of SPARC International Inc. "VxWorks" and "Tornado" are registered trademarks of Wind River Systems Inc. "Xilinx" is a registered trademark of Xilinx.
All other trademarks and registered trademarks are the property of their respective owners.

Part Information

Part Number: SNP-MAN-JPG-LIB

Version v4.0.1 September 1999

Printed in the United Kingdom.

Contact Details

Europe & ROW	Web	www.datacell.co.uk
	Sales	info@datacell.co.uk
	Support	techsupport@datacell.co.uk
USA	Web	www.datacell.com
	Sales	info@datacell.com
	Support	techsupport@datacell.com

Head Office:
DataCell Limited.
Falcon Business Park, 40 Ivanhoe Road,
Finchampstead, Berkshire, RG40 4QQ, UK

Tel +44 (0) 1189 324324
Fax +44 (0) 1189 324325

Table of Contents

Introduction

This document describes the "CRUNCH" library of functions. These functions allow the compression and decompression of images using a "Crunch" JPEG compression card. The functions are able to process images from files, memory, and also directly from a Snapper module plugged onto a Crunch board. (e.g. Snapper-16 -a composite/S-Video image acquisition module.)

Function Overview

INITIALIZATION

The Crunch compression hardware resides on the Bus Interface Board and is initialized by the Bus Interface Library function call *BASE_create*. Please refer to the Bus Interface Board Library for more details.

Prior to compressing an image, the desired quality factor can be set using *CRUNCH_set_Quality_factor* or *CRUNCH_set_Quantization_factor* (only one of these functions needs to be used). If neither of these functions is used, the default quality factor of 32 will be used. Refer to the manual pages on these functions for a description of what this actually means.

COMPRESSION / DECOMPRESSION FUNCTIONS

The compression and decompression functions operate on TMG image handles and require the TMG imaging processing library. The image handle references an image structure that contains image data, along with additional information such as image width, height, depth, colourmap etc. The functions allow images to be processed as a whole, or in a series of strips. Strip processing is required where memory is at a premium (e.g. 16 bit real mode MS-DOS), or where very large images are being processed. For a detailed description of strip processing, and the creation of images, see the "TMG Library Programmer's Manual".

The functions can operate on image data located either in memory, in a file or from a Snapper acquisition module.

In order to allow the greatest flexibility, two levels of user compression/decompression routines are provided. At the highest level, *CRUNCH_compress_image_to_image, CRUNCH_decompress_image_to_image* and *CRUNCH_compress_from_snapper* perform complete operations. These functions allow an image to be compressed or decompressed with just a single function call.

CRUNCH_compress_image_to_image reads raw data from the source image, and writes compressed JPEG data to the destination image. Internally, this function operates in strips, the strip height (i.e. number of lines) being determined by the *lines_this_strip* parameter of the source image. The source and destination images can be to/from memory or disk.

Similarly, *CRUNCH_decompress_image_to_image* reads compressed data from a source image and writes raw data to a destination image.

CRUNCH_compress_from_snapper reads raw data directly from a Snapper module, and writes compressed JPEG data to a destination JPEG image (in memory or to directly to disk).

At a lower level to the above functions, *CRUNCH_compress* and *CRUNCH_decompress* allow the processing of a single strip. Again, the strip height is determined by the *lines_this_strip* parameter of the input image. For more information and examples of strip processing, refer to the TMG Library Programmer's Manual and the example source code provided as part of the Snapper SDK.

CRUNCH_sequence_record records a motion JPEG sequence to memory or to file.

CRUNCH_sequence_replay replays a motion JPEG sequence from memory or file.

Error Returns

All of the CRUNCH library functions return a type called *Terr*. This is a 32 bit unsigned integer, with the bit positions defined as follows:

31 to 24	Hardware ident and revision of the Bus Interface Board that the handle refers to. If there is no error this field is zero. The upper 5 bits of this byte refer to the ident and the lower 3 bits the revision level of the board.
23 to 16	This contains an error number, otherwise 0 if no error.
15 to 0	Function return value.

If a function call is successful, it returns *ASL_OK* (which is defined as 0) or the requested parameter. If an error occurs, an error number is returned in bits 23 to 16 along with the library identifier in bits 31 to 24. See the "Snapper Error Handling Programmer's Manual" in the Developer's Guide section of the Snapper Developer's Manual for more details on error returns.

Sample Applications

The following four programs show how to compress/decompress an image to/from a JPEG file. The first compression example assumes a 32 bit DOS extender (the Symantec DOSX model is supported) and the second compression example shows how to use real mode MS-DOS (for example Microsoft C with no DOS extender). Then there are two decompression examples - again one using assuming an MS- DOS extender and one assuming real mode MS-DOS. Note that the examples illustrating how to use real mode MS-DOS work equally well with a MS-DOS extender (but uses significantly less dynamic memory). As with all sample code in this manual, error handling has been omitted for clarity, but apart from not handling errors cleanly these are usable programs. More detailed examples can be found on the Snapper SDK disks.

```
#include  <asl_inc.h>

int main(ui16 argc, char** argv)
{
   Thandle Hbase;                   /* Handle to Crunch hardware */
   Thandle Hin_image, Hjpeg_image;  /* Input and output images   */

   /* Initialize Crunch baseboard */
   Hbase = ASL_get_ret(BASE_create(BASE_AUTO));
   Hin_image = TMG_image_create();
   Hjpeg_image = TMG_JPEG_image_create();

   TMG_image_set_infilename(Hin_image, “input.tif”);
   TMG_image_set_outfilename(Hjpeg_image, “output.jpg”);

   TMG_image_set_parameter(Hin_image, TMG_HEIGHT, TMG_AUTO_HEIGHT);
   TMG_image_read(Hin_image, TMG_NULL, TMG_RUN);

   /* Compress the image in memory to a JPEG image also in memory */
   CRUNCH_compress(Hbase, Hin_image, Hjpeg_image, TMG_RUN);
   TMG_JPEG_write_file(Hjpeg_image, TMG_RUN);

   /* Free memory and exit */
   BASE_destroy(BASE_ALL_HANDLES);
   TMG_image_destroy(TMG_ALL_HANDLES);
}
```

The following example performs exactly the same function as the previous one, but processes the image in strips, thus dramatically reducing the amount of dynamic memory required. Note that this code will work in either real mode MS-DOS or extended mode MS-DOS.

```
#include  <asl_inc.h>

int main(ui16 argc, char** argv)
{
   Thandle Hbase;                   /* Handle to Crunch hardware */
   Thandle Hin_image, Hjpeg_image;  /* Input and output images   */

   /* Initialize Crunch baseboard */
   Hbase = ASL_get_ret(BASE_create(BASE_AUTO));
   Hin_image = TMG_image_create();
   Hjpeg_image = TMG_JPEG_image_create();

   TMG_image_set_infilename(Hin_image, “input.tif”);
   TMG_image_set_outfilename(Hjpeg_image, “output.jpg”);

   /* Read the image's dimensions */
   TMG_image_set_parameter(Hin_image, TMG_LINES_THIS_STRIP, 0);
   if ( TMG_image_read(Hin_image, TMG_NULL, TMG_RUN) != ASL_OK ) {
      printf("\nFailed to open file\n");
      exit(0);
   }
   TMG_image_read(Hin_image, TMG_NULL, TMG_RESET);

   /* We’ll process 8 lines at a time */
   TMG_image_set_parameter(Hin_image, TMG_LINES_THIS_STRIP, 8);

   /* Compress the image from file to file */
   CRUNCH_compress_image_to_image(Hbase, Hin_image, Hjpeg_image, TMG_FILE,
            TMG_FILE);

   /* Free memory and exit */
   BASE_destroy(BASE_ALL_HANDLES);
   TMG_image_destroy(TMG_ALL_HANDLES);
}
```

The next example shows how to decompress an image in memory and requires an MS-DOS extender (Symantec DOSX).

```
#include  <asl_inc.h>

int main(ui16 argc, char** argv)
{
   Thandle Hbase;                   /* Handle to Crunch hardware */
   Thandle Hjpeg_image, Hout_image; /* Input and output images   */

   /* Initialize Crunch baseboard */
   Hbase = ASL_get_ret(BASE_create(BASE_AUTO));
   Hout_image = TMG_image_create();
   Hjpeg_image = TMG_JPEG_image_create();

   TMG_image_set_infilename(Hjpeg_image, “input.jpg”);
   TMG_image_set_outfilename(Hout_image, “output.tif”);

   if (TMG_JPEG_read_file(Hjpeg_image) != ASL_OK) {
      printf(“DECOMP: Failed to open JPEG file\n”);
      exit(0);
   }

   /* Process the whole image in one strip */
   TMG_image_set_parameter( Hjpeg_image, TMG_LINES_THIS_STRIP,
                    TMG_image_get_parameter(Hjpeg_image, TMG_HEIGHT) );

   /* Decompress the JPEG image in memory to raw data in memory */
   CRUNCH_decompress(Hcrunch, Hjpeg_image, Hout_image, TMG_RUN);

   /* Now save the image as a TIFF file */
   TMG_image_write(Hout_image, TMG_NULL, TMG_TIFF, TMG_RUN);

   /* Free memory and exit */
   BASE_destroy(BASE_ALL_HANDLES);
   TMG_image_destroy(TMG_ALL_HANDLES);
}
```

This final example shows how to decompress a JPEG image from file to file. This code will work in either real mode MS-DOS or extended mode MS-DOS and uses significantly less dynamic memory than the previous example.

```
#include  <asl_inc.h>

int main(ui16 argc, char** argv)
{
   Thandle Hbase;                   /* Handle to Crunch hardware */
   Thandle Hjpeg_image, Hout_image; /* Input and output images   */

   /* Initialize Crunch baseboard */
   Hbase = ASL_get_ret(BASE_create(BASE_AUTO));
   Hout_image = TMG_image_create();
   Hjpeg_image = TMG_JPEG_image_create();

   TMG_image_set_infilename(Hjpeg_image, “input.jpg”);
   TMG_image_set_outfilename(Hout_image, “output.tif”);

   /* Process 8 lines at a time to reduce memory requirements */
   TMG_image_set_parameter(Hjpeg_image, TMG_LINES_THIS_STRIP, 8);

   /* Decompress from the JPEG file to a TIFF file */
   CRUNCH_decompress_image_to_image(Hbase, Hjpeg_image, Hout_image, TMG_FILE,
            TMG_TIFF);

   /* Free memory and exit */
   BASE_destroy(BASE_ALL_HANDLES);
   TMG_image_destroy(TMG_ALL_HANDLES);
}
```

Note that JPEG software is provided in the TMG library and that the above Crunch functions may be replaced by “*TMG_JPEG*” ones. See the TMG Programmer’s Manual for more details.

Function List

CRUNCH_compress

CRUNCH_compress_from_snapper

CRUNCH_compress_image_to_image

CRUNCH_decompress

CRUNCH_decompress_image_to_image

CRUNCH_sequence_record

CRUNCH_sequence_replay

CRUNCH_set_Quality_factor

CRUNCH_set_Quantization_factor

The functions are described in alphabetical order in the following pages.

CRUNCH_compress_from_snapper

USAGE

Terr CRUNCH_compress_from_snapper(Thandle Hbase, Thandle Hjpeg_image, ui16 format)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Hjpeg_image	Handle to JPEG compressed image.
format	Either *TMG_MEMORY* or *TMG_FILE.*

DESCRIPTION

This function compresses the image already captured in a Snapper-16 or Snapper-24 module using the JPEG hardware referenced by *Hbase*. Raw image data is read from the Snapper module, compressed and written to *Hjpeg_image*. If the output format, *format,* is set to *TMG_MEMORY*, the compressed data is written to memory. If *format* is set to *TMG_FILE* it is written directly to a JPEG file with the name specified by a call to *TMG_image_set_outfilename* with *Hjpeg_image* as the image handle.

If the sub-sampling mode of the Snapper is set to *SUB_X1_FIELD_DUPLICATE*, only single fields will be compressed. Also the image height will be halved and the flag, *TMG_HALF_ASPECT*, will be set.

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.
ASLERR_NOT_SUPPORTED	The baseboard does not contain any compression hardware, ie it is NOT an ISA-JPG.

EXAMPLES

The following code will capture an image from Snapper-16 and then save the image as a JPEG file.

```
.
SNP16_capture(Hsnapper);
TMG_image_set_outfilename(Hjpeg_image, “image.jpg”);
CRUNCH_compress_from_snapper(Hbase, Hjpeg_image, TMG_FILE);
.
```

BUGS / NOTES

The image to be compressed must have a width which is exactly divisible by 16 for colour images and 8 for grayscale images, and a height that is exactly divisible by 8. Also the minimum image size is 128 x 8.

The JPEG images are generated using the “default” Huffman tables as suggested in the JPEG specification.

This function has the side effect of setting the image parameter *TMG_LINES_THIS_STRIP* to the height of the image.

SEE ALSO

CRUNCH_compress, *CRUNCH_sequence_record*.

CRUNCH_compress

USAGE

Terr CRUNCH_compress(Thandle Hbase, Thandle Himage, Thandle Hjpeg_image, ui16 TMG_action)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Himage	Handle to a raw (uncompressed) image.
Hjpeg_image	Handle to a JPEG compressed image.
TMG_action	Either *TMG_RUN* for normal operation or *TMG_RESET* to abort.

DESCRIPTION

This function compresses a single image strip using the JPEG hardware referenced by *Hbase*. If the function is called with *TMG_action* set to *TMG_RUN*, raw image data is read from *Himage*, and compressed JPEG data written to *Hjpeg_image*. The strip size is determined by the *lines_this_strip* parameter of *Himage*. If the function is called with *TMG_action* set to *TMG_RESET* the compression process is aborted and local static (internal) variables are reset. *TMG_RESET* is rarely needed.

This function is called by *CRUNCH_compress_image_to_image.*

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.
ASLERR_NOT_SUPPORTED	This can be for one of the following reasons: The baseboard does not contain any compression hardware, ie it is NOT an ISA-JPG. The image height and width are not multiples of 8 pixels The image format is not *TMG_Y8*, *TMG_YUV422* or *TMG_RGB24*.

EXAMPLES

See the Sample Applications section at the beginning of this manual.

BUGS / NOTES

The image to be compressed must have a width which is exactly divisible by 16 for colour images and 8 for grayscale images, and a height that is exactly divisible by 8. The width needs to be at least 128 pixels for colour images and 64 pixels for mono images; and at least 8 lines high.

The JPEG images are generated using the "default" Huffman tables as suggested in the JPEG specification.

SEE ALSO

CRUNCH_compress_from_snapper, *CRUNCH_set_Quality_factor*, *CRUNCH_set_Quantization_factor*.

CRUNCH_compress_image_to_image

USAGE

Terr CRUNCH_compress_image_to_image(Thandle Hbase, Thandle Himage, Thandle Hjpeg_image, ui16 in_format, ui16 out_format)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Himage	Handle to raw image.
Hjpeg_image	Handle to compressed image.
in_format	*TMG_MEMORY* or *TMG_FILE.*
out_format	*TMG_MEMORY* or *TMG_FILE.*

DESCRIPTION

This is a convenient wrapper function for *CRUNCH_compress* that compresses a complete image using the JPEG hardware referenced by *Hbase*. Raw image data is read from *Himage*, compressed and written to *Hjpeg_image*. If the input format, *in_format*, is set to *TMG_MEMORY*, raw image data is read from memory (from *Himage*). If *in_format* is set to *TMG_FILE*, it is read from the file associated with *Himage* (i.e. set using *TMG_image_set_infilename*). Similarly, If the output format, *out_format*, is set to *TMG_MEMORY*, compressed data is written to memory (in *Hjpeg_image*). If *out_format* is set to *TMG_FILE* it is written directly to the JPEG file referenced by *Hjpeg_image* (i.e. set using *TMG_image_set_outfilename*).. If the *lines_this_strip* parameter of *Himage* is less than the total image height, then compression is performed in strips. The *lines_this_strip* parameter is set using *TMG_image_set_parameter*.

For DOS real mode applications, *lines_this_strip* should typically be set to 8. This is due to the memory limitations of real mode DOS. This has very little affect on the compression speed. See the TMG Library Programmer's Manual for more details on strip processing.

This function is a convenient way of compressing from file to file with just one call.

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.
ASLERR_NOT_SUPPORTED	The baseboard does not contain any compression hardware, ie it is NOT an ISA-JPG.

EXAMPLES

See the Sample Applications section at the beginning of this manual.

BUGS / NOTES

The strip size must be a multiple of 8.

The image to be compressed must have a width which is exactly divisible by 16 for colour images and 8 for grayscale images, and a height that is exactly divisible by 8. The width needs to be at least 128 pixels for colour images and 64 pixels for mono images; and at least 8 lines high.

The JPEG images are generated using the "default" Huffman tables as suggested in the JPEG specification.

SEE ALSO

CRUNCH_compress, *CRUNCH_compress_from_snapper*, *CRUNCH_set_Quality_factor*, *CRUNCH_set_Quantization_factor*.

CRUNCH_decompress

USAGE

Terr CRUNCH_decompress(Thandle Hbase, Thandle Hjpeg_image, Thandle Himage, ui16 TMG_action)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Hjpeg_image	Handle to compressed JPEG image.
Himage	Handle to raw (uncompressed) image.
TMG_action	Either *TMG_RUN* for normal operation or *TMG_RESET* to abort.

DESCRIPTION

This function decompresses a single image strip using the JPEG hardware referenced by *Hbase*. If the function is called with *TMG_action* set to *TMG_RUN*, compressed image data is read from *Hjpeg_image*, and raw data written to *Himage*. The strip size is determined by the *lines_this_strip* parameter of *Hjpeg_image*. If the function is called with *TMG_action* set to *TMG_RESET*, the decompression is aborted and internal static variables are reset.

This function is called by *CRUNCH_decompress_image_to_image.*

For colour images, the JPEG image may be decompressed to YUV data or RGB data depending on the pixel format set in *Himage* (see *TMG_image_set_parameter* with *TMG_PIXEL_FORMAT*). By default the decompressed image will have the pixel format *TMG_RGB24*, but if the pixel format is set to *TMG_YUV422* prior to calling this function, the pixel format of the decompressed image will be *TMG_YUV422*.

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.
ASLERR_NOT_SUPPORTED	This can be for one of the following reasons: The baseboard does not contain any compression hardware, ie it is NOT an ISA-JPG. The image height and width are not multiples of 8 pixels

EXAMPLES

See the Sample Applications section at the beginning of this manual.

BUGS / NOTES

The strip size must be a multiple of 8.

Only JPEG images with the "default" Huffman tables as suggested in the JPEG specification are supported.

SEE ALSO

CRUNCH_decompress_image_to_image.

CRUNCH_decompress_image_to_image

USAGE

Terr CRUNCH_decompress_image_to_image(Thandle Hbase, Thandle Hjpeg_image, Thandle Himage, ui16 in_format, ui16 out_format)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Hjpeg_image	Handle to compressed image.
Himage	Handle to raw image.
in_format	*TMG_MEMORY* or *TMG_FILE.*
out_format	*TMG_MEMORY* or a file type: *TMG_TIFF*, *TMG_TGA*, *TMG_EPS*, *TMG_BMP etc.*

DESCRIPTION

This function decompresses a complete image using the JPEG hardware referenced by *Hbase.* Compressed image data is read from *Hjpeg_image*, and raw image data written to *Himage.* If the parameter, *in_format*, is set to *TMG_MEMORY*, compressed image data is read directly from memory (referenced by *Hjpeg_image*). If *in_format* is set to *TMG_FILE,* the JPEG data is read from the JPEG/JFIF file associated with *Hjpeg_image*. Similarly, if the parameter, *out_format*, is set to *TMG_MEMORY*, raw image data is written to memory (in *Himage*). If *out_format* is set to a file type, for example *TMG_TIFF*, it is written directly to the file referenced by *Himage* in that format. If the *lines_this_strip* parameter of *Hjpeg_image* is less than the total image height, the decompression is performed in strips.

When decompressing colour images to memory, the output pixel format may be YUV or RGB depending on the pixel format set in *Himage* (see *TMG_image_set_parameter* with *TMG_PIXEL_FORMAT*). By default the decompressed image will have the pixel format *TMG_RGB24*, but if the pixel format is set to *TMG_YUV422* prior to calling this function, the pixel format of the decompressed image will be *TMG_YUV422*.

For real mode DOS applications, *lines_this_strip* should typically be set to 8. This is due to the memory limitations of real mode DOS. This has very little affect on the decompression speed. See the TMG Library Programmer's Manual for more details on strip processing.

This function is a convenient way of decompressing from file to file with just one call.

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.
ASLERR_NOT_SUPPORTED	The baseboard does not contain any compression hardware, ie it is NOT an ISA-JPG.

EXAMPLES

See the Sample Applications section at the beginning of this manual.

BUGS / NOTES

The strip size must be a multiple of 8.

Only JPEG images with the “default” Huffman tables as suggested in the JPEG specification are supported.

SEE ALSO

CRUNCH_decompress, *CRUNCH_compress_image_to_image*.

CRUNCH_sequence_record

USAGE

Terr CRUNCH_sequence_record(Thandle Hbase, Thandle Hjpeg_image, ui32 length, ui16 format, ui16 TMG_action)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Hjpeg_image	Handle to compressed image sequence.
length	Length of sequence in frames.
format	Either *TMG_MEMORY* or *TMG_FILE*.
TMG_action	*TMG_START, TMG_RUN or TMG_RESET.*

DESCRIPTION

This function record a sequence of frames from Snapper-24 or Snapper-16 and saves them as a compressed motion JPEG sequence referenced by *Hjpeg_image*. The JPEG data may be streamed either to disk or to memory using the *format* parameter. This function does not reset the JPEG decompression hardware between frames and is therefore faster than repeatedly using *CRUNCH_compress* on a sequence of images. Memory must first be allocated as shown in the example below.

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.
ASLERR_NOT_SUPPORTED	The baseboard does not contain any compression hardware, ie it is NOT an ISA-JPG.

EXAMPLES

The following code records a motion JPEG sequence to memory:

```
/* malloc a megabyte for a sequence - in Hjpeg_image */
Pmem = (IM_UI8*) MALLOC( (long) 100000 );
if ( Pmem == NULL ) {
   printf(“No memory available!”);
   exit(0);
}

TMG_image_set_ptr(Hjpeg_image, TMG_JPEG_DATA, (void*) Pui8_1);
/* stop the library from freeing memory it hasn’t allocated */
TMG_image_set_flags(Hjpeg_image, TMG_LOCKED, TRUE);
.
/* record 32 frames to memory */
CRUNCH_sequence_record(Hcrunch, Hjpeg_image, 32, TMG_MEMORY, TMG_RUN);
.
/* note the application must free the memory */
FREE(Pmem);
TMG_image_set_ptr(Hjpeg_image, TMG_JPEG_DATA, (void*) NULL);
TMG_image_set_flags(Hjpeg_image, TMG_LOCKED, FALSE);
BASE_destroy(BASE_ALL_HANDLES);
TMG_image_destroy(TMG_ALL_HANDLES);
```

This next example records a motion JPEG sequence directly to disk.

```
/*
 * We don’t need to allocate any memory - a single frame
 * of workspace memory will be malloc’ed internally.
 */

/* record 32 frames to disk */
TMG_image_set_outfilename(Hjpeg_image, “seq.jpg”);
CRUNCH_sequence_record(Hcrunch, Hjpeg_image, 32, TMG_FILE, TMG_RUN);
```

For a detailed example please refer to the file “seq.c” on the Snapper SDK disks.

BUGS / NOTES

This function is supported under all operating systems/environments apart from real mode DOS because of the memory limitations. Also this function always compresses complete images and does not compress the image in strips.

The JPEG data format is exactly the same as “normal” JPEG/JFIF files except the data consists of multiple frames with restart markers (FF D8 hex) between each frame.

SEE ALSO

TMG_sequence_set_start_frame, *CRUNCH_sequence_replay*.

CRUNCH_sequence_replay

USAGE

Terr CRUNCH_sequence_replay(Thandle Hbase, Thandle Hjpeg_image, Thandle Hout_image, ui16 subsample, ui16 TMG_action)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Hjpeg_image	Handle to compressed image sequence.
Hout_image	Handle to raw image.
subsample	Replay sub-sampling ratio.
TMG_action	*TMG_START, TMG_RUN or TMG_RESET.*

DESCRIPTION

This function decompresses a single image from the JPEG sequence referenced by *Hjpeg_image* to *Hout_image*, each time the function is called. This function does not reset the JPEG decompression chip between frames and is therefore faster than repeatedly using *CRUNCH_decompress* on a sequence of images. It also decompresses directly to the *TMG_RGB16* format, suitable for direct display to the host's graphics card.

The function is first called with *TMG_action* set to *TMG_START* - this initialises the hardware and pre-fills the decompression pipeline. Subsequent calls that decompress a single image are made with this parameter set to *TMG_RUN.* Having finished replaying the sequence, the JPEG hardware should be reset by calling the same function with *TMG_action* set to *TMG_RESET.*

The parameter, *subsample,* can be used to sub-sample the image on replay thus saving a memory to memory copy (i.e. the case if *TMG_image_subsample* was used).

The sequence can be single stepped or played from any location by using the function *TMG_JPEG_sequence_set_start_frame.*

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.
ASLERR_NOT_SUPPORTED	The baseboard does not contain any compression hardware, ie it is NOT an ISA-JPG.

EXAMPLES

The following code replays a motion JPEG sequence from memory to a RGB16 format display:

```
TMG_JPEG_sequence_set_start_frame(Hjpeg_image, 1);
CRUNCH_sequence_replay(Hcrunch, Hjpeg_image, Himage1, 1, TMG_START);

for (n = 0; n < 32; n++)
{
   CRUNCH_sequence_replay(Hcrunch, Hjpeg_image, Himage1, 1, TMG_RUN);
   TMG_display_image(Hdisplay, Himage1, TMG_RUN);
}
CRUNCH_sequence_replay(Hcrunch, Hjpeg_image, Himage1, 1, TMG_RESET);
```

This next example replays a motion JPEG sequence directly from disk.

```
/* Read/open in input file, then replay as before */
TMG_image_set_infilename(Hjpeg_image, "seq.jpg");
TMG_JPEG_open_file(Hjpeg_image);
TMG_JPEG_sequence_calc_length(Hjpeg_image); /* works on file */
/* Could do a "TMG_JPEG_read_file(Hjpeg_image);" */

TMG_JPEG_sequence_set_start_frame(Hjpeg_image, 1);
CRUNCH_sequence_replay(Hcrunch, Hjpeg_image, Himage1, 1, TMG_START);

for (n = 0; n < 32; n++)
{
   CRUNCH_sequence_replay(Hcrunch, Hjpeg_image, Himage1, 1, TMG_RUN);
   TMG_display_image(Hdisplay, Himage1, TMG_RUN);
}

CRUNCH_sequence_replay(Hcrunch, Hjpeg_image, Himage1, 1, TMG_RESET);
```

For a detailed example please refer to the file "seq.c" on the Snapper SDK disks.

BUGS / NOTES

This function is supported under all operating systems/environments apart from real mode DOS because of the memory limitations. Also this function always decompresses complete images and does not decompress the image in strips.

The JPEG data format is exactly the same as "normal" JPEG/JFIF files except the data consists of multiple frames with restart markers (FF D8 hex) between each frame.

SEE ALSO

TMG_sequence_set_start_frame, *CRUNCH_sequence_record*.

CRUNCH_set_Quality_factor

USAGE

Terr CRUNCH_set_Quality_factor(Thandle Hbase, ui16 Q_factor)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Q_factor	An integer representing image quality after compression.

DESCRIPTION

This function sets the JPEG quality factor for compression. It is not used in decompression. It represents the quality of the compressed image. A lower quality factor means lower image quality and higher compression ratio, and vice-versa.

The range of the quality factor is from 1 to 400, although numbers above 80 will give very little improved quality but with low (typically 6:1) compression ratios.

The default quality factor of 32 is used if this function (or *CRUNCH_set_Quantization_factor*) is not called. This results in an image which the JPEG Specification describes as "usually nearly indistinguishable from the original".

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.

EXAMPLES

The following code sets the quality factor:

```
..
CRUNCH_set_Quality_factor(Hbase, 16);
..
```

BUGS / NOTES

There are no known bugs.

The quality factor is normalised to 16, which is common practice in JPEG software packages. This means that when set to 16, the luminance and chrominance quantization tables are identical to those in the JPEG Specification.

Note that quality factor and quantization factor are alternative ways of controlling the compression ratio. The relationship is:

Quality factor = (50 / Quantization Factor) x 16

Both *CRUNCH_set_Quality_factor* and *CRUNCH_set_Quantization_factor* are provided to allow application developers to work with the most convenient parameter in their application.

SEE ALSO

CRUNCH_set_Quantization_factor.

CRUNCH_set_Quantization_factor

USAGE

Terr CRUNCH_set_Quantization_factor(Thandle Hbase, ui16 Q_factor)

ARGUMENTS

Hbase	Handle to Crunch JPEG hardware.
Q_factor	JPEG quantization factor.

DESCRIPTION

This function sets the JPEG quantization factor for compression. It is not used in decompression. It is used to generate the quantization table which defines the number of quantization levels at which the luminance and chrominance frequencies are quantized to. In simple terms, a higher quantization factor means lower image quality and higher compression ratio, and vice-versa.

The range of the quantization factor is from 8 to 800, although numbers below 10 will give very little improved quality but with low (typically 6:1) compression ratios.

The default quantization factor of 25 is used if this function (or *CRUNCH_set_Quality_factor*) is not called. This results in an image which the JPEG Specification describes as “usually nearly indistinguishable from the original”.

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The ISA-JPG handle is invalid.

EXAMPLES

The following code sets the quantization factor:

```
..
CRUNCH_set_Quantization_factor(Hbase, 100);
..
```

BUGS / NOTES

There are no known bugs.

The quantization factor is normalised to 50, which is consistent with other JPEG (hardware) systems. This means that when set to 50, the luminance and chrominance quantization tables are identical to those in the JPEG Specification.

Note that quality factor and quantization factor are alternative ways of controlling the compression ratio. The relationship is:

Quality factor = (50 / Quantization Factor) x 16

Both *CRUNCH_set_Quality_factor* and *CRUNCH_set_Quantization_factor* are provided to allow application developers to work with the most convenient parameter in their application.

SEE ALSO

CRUNCH_set_Quality_factor.