SNAPPER-16 Library

(for Snapper-16 and Snapper-PCI16)

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

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-SNP16-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 manual describes the Snapper-16 function library. These functions allow the capture of video images, using a Snapper-16 module and one of a number of different host hardware platforms, and are independent of the host hardware platform. This manual also applies to the Snapper-PCI16 product, which are single board combinations of a PCI-BIB baseboard and a Snapper-16 module.

Snapper-16 is referred to by a unique handle of type *Thandle* (a 32 bit unsigned integer). It is used by all the software to identify a particular Snapper board and its associated data structures. This handle is automatically generated when a Bus Interface Board detects it has a Snapper module fitted.

Function Overview

The functions are split into five sections - Initialization, Image Capture, Configuration, Parameter Readback and Miscellaneous.

INITIALIZATION FUNCTIONS

Four functions are provided which configure Snapper-16 for the most common applications: *SNP16_configure_gray_60Hz*, *SNP16_configure_gray_50Hz* and *SNP16_configure_NTSC*, *SNP16_configure_PAL*.

IMAGE CAPTURE

The image capture functions control capture of images and provide functions to test the capture status.

SNP16_capture and *SNP16_start_capture* control the capture of video data into Snapper-16's video memory. *SNP16_read_video_data* reads the data from Snapper-16's video memory into an image structure in host memory. This image structure is set up by *SNP16_set_image*. *SNP16_reset_read_pointer* allows the data to be read again. Capture status is indicated by *SNP16_is_capture_complete* and *SNP16_is_trigger_started*.

CONFIGURATION

These functions control the configuration of the Snapper-16.

The video data can be sub-sampled and horizontally filtered by calling *SNP16_set_subsample*. Image capture can be triggered from external hardware by using the *SNP16_set_ext_trigger*, *SNP16_set_ext_polarity* and *SNP16_set_trigger_timeout* functions. Selected regions of the image can be captured by a call to *SNP16_set_ROI*.

The video source to be digitized is controlled by *SNP16_set_video_source* and *SNP16_set_input_mode*. The format in which video data is stored in memory is controlled by *SNP16_set_format* and *SNP16_set_mono7_offset*.

SNP16_set_mode_vcr alters the phase locked loop (PLL) time constant to allow stable locking to video cassette recorders. *SNP16_set_vertical_mode* can be used to configure Snapper-16 to acquire from asynchronous reset cameras.

The remaining configuration functions are called by *SNP16_configure_xxx* during initialisation and are generally not called by applications. These functions are: *SNP16_set_chrominance*, *SNP16_set_clamp_start*, *SNP16_set_clamp_stop*, *SNP16_set_sync_start*, *SNP16_set_sync_stop*, *SNP16_set_decode_mode*, *SNP16_set_deinterlace*, *SNP16_set_filter* and *SNP16_set_hue*.

PARAMETER READBACK FUNCTIONS

Most of these functions are intended to avoid the need for an application to keep shadow copies of Snapper-16 settings. These are *SNP16_get_chrominance*, *SNP16_get_decode_mode*, *SNP16_get_format*, *SNP16_get_hue*, *SNP16_get_parameter*, *SNP16_get_ROI* and *SNP16_get_subsample*.

SNP16_get_ROI_max returns the maximum allowable ROI size for the camera in use, and *SNP16_get_property* returns hardware and firmware information about the Snapper-16. The final readback functions are *SNP16_get_ID* and *SNP16_get_rev* which return the hardware revision information of the Snapper-16 in use.

MISCELLANEOUS FUNCTIONS

Four functions can be used to test the video standard: *SNP16_is_50Hz*, *SNP16_is_60Hz*, *SNP16_is_locked* and *SNP16_is_colour*.

Error Returns

All of the Snapper-16 library functions return a *Terr* apart from several Boolean functions. *Terr* is a 32 bit unsigned integer, with the bit positions defined as follows:

31 to 24 Hardware identifier/revision (returned on error, otherwise 0 is returned). This is used to allow a top level calling function to determine the library in which the error occurred, and is actually read from the hardware itself.

Clearing bits 26 to 24 leaves the hardware identifier, which is 0xA0 (#defined as *SNP16_ID*).

Bits 26 to 24 give the hardware revision level. Initial Snapper-16s have the value 0x00.

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 hardware or 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 Application

The following code is a minimal program to capture an image from a PAL camera using a Snapper-16 and save it to a file. As with all sample code in this manual, error handling has been omitted for clarity. For examples of how to use the built in error messaging, refer to the "Snapper Error Handling" Programmer's Manual. The Snapper SDK includes sample applications, both as executables and as source code, which provide a useful reference of 'real' code and are probably the best starting point for developing custom applications. For examples of how to display images under different operating systems see the examples in the TMG Library Programmer's Manual.

```
#include <asl_inc.h>

int main(ui16 argc, char** argv)
{
   Thandle Hsnp16;       /* Handle to Snapper-16 */
   Thandle Hbase;        /* Handle to baseboard */
   Thandle Hvid_image;   /* Handle to YUV image */
   Thandle Hrgb_image;   /* Handle to RGB image */

   /* Initialize baseboard and Snapper module */
   Hbase = ASL_get_ret(BASE_create(BASE_AUTO));
   Hsnp16 = BASE_get_parameter(Hbase, BASE_MODULE_HANDLE);
   Hvid_image = TMG_image_create();
   Hrgb_image = TMG_image_create();

   /* Set required Snapper mode */
   SNP16_configure_PAL(Hsnp16);

   /* Set up image parameters */
   SNP16_set_image(Hsnp16, Hvid_image);

   /* Capture image and save it as a TIFF file */
   SNP16_capture(Hsnp16);
   SNP16_read_video_data(Hsnp16, Hvid_image, TMG_RUN);
   TMG_image_set_outfilename(Hvid_image, "image.tif");
   TMG_image_convert(Hvid_image, Hrgb_image, TMG_RGB24, 0, TMG_RUN);
   TMG_image_write(Hrgb_image, NULL, TMG_TIFF, TMG_RUN);

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

Function List

INITIALIZATION FUNCTIONS

IMAGE CAPTURE FUNCTIONS

CONFIGURATION FUNCTIONS

PARAMETER READ BACK FUNCTIONS

MISCELLANEOUS FUNCTIONS

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

SNP16_capture

USAGE

Terr SNP16_capture(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function digitizes the next field or frame of video data into Snapper-16's on-board video memory. Snapper-16 must be initialized with the required video source, region of interest etc, prior to calling this function. The function does not return until Snapper-16 has completed capturing the required data. When external trigger is enabled the function returns after a timeout period if no trigger occurs.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

If timeouts are required for capture it is recommended that they are controlled within the application by using the *SNP16_start_capture* function together with the clock functions available within the operating system in use.

SEE ALSO

SNP16_start_capture, *SNP16_read_video_data*, *SNP16_set_trigger_timeout*, *SNP16_is_trigger_started*.

SNP16_configure_gray_60Hz, SNP16_configure_gray_50Hz

USAGE

Terr SNP16_configure_gray_60Hz(Thandle Hsnp16)

Terr SNP16_configure_gray_50Hz(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

SNP16_configure_gray_60Hz configures Snapper-16 to acquire 60Hz (EIA RS-170/RS-170A) monochrome video with a full screen ROI (region of interest), i.e. 640 pixels wide by 480 lines high.

SNP16_configure_gray_50Hz configures Snapper-16 to acquire 50Hz (CCIR) monochrome video with a full screen ROI (region of interest), i.e. 768 pixels wide by 576 lines high.

The TMG pixel format of the resulting Himage is set to TMG_Y8.

The functions make calls to *SNP16_set_decode_mode*, *SNP16_set_chrominance*, *SNP16_set_input_mode*, *SNP16_set_ROI* and *SNP16_set_format*.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_configure_NTSC, *SNP16_configure_PAL*.

SNP16_configure_NTSC, SNP16_configure_PAL

USAGE

Terr SNP16_configure_NTSC(Thandle Hsnp16)

Terr SNP16_configure_PAL(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

SNP16_configure_NTSC configures Snapper-16 for 60Hz colour image capture. This is in composite NTSC decode mode and with a full screen ROI (region of interest), i.e. 640 pixels wide by 480 lines high.

SNP16_configure_PAL configures Snapper-16 for 50Hz colour image capture. This is in composite PAL decode mode and with a full screen ROI (region of interest), i.e. 768 pixels wide by 576 lines high.

The TMG library pixel format of the resulting Himage is set to TMG_YUV422.

The functions make calls to *SNP16_set_decode_mode*, *SNP16_set_chrominance*, *SNP16_set_input_mode*, *SNP16_set_ROI* and *SNP16_set_format*. *SNP16_set_input_mode* should be called directly if YC video rather than composite video is in use.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_configure_gray_60Hz, *SNP16_configure_gray_50Hz*.

SNP16_get_chrominance

USAGE

Terr SNP16_get_chrominance(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function returns the current chrominance gain setting.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

The <chrominance> is returned in the lower 8 bits, if successful.

There are no known bugs.

SEE ALSO

SNP16_set_chrominance, *SNP16_set_hue*.

SNP16_get_decode_mode

USAGE

Terr SNP16_get_decode_mode(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function returns the current colour decoder mode. This is one of *SNP16_DECODE_PAL*, *SNP16_DECODE_NTSC* or *SNP16_DECODE_SECAM*.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

The <decode_mode> is returned in the lower 16 bits, if successful.

There are no known bugs.

SEE ALSO

SNP16_set_decode_mode, *SNP16_get_format*.

SNP16_get_format

USAGE

Terr SNP16_get_format(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function returns Snapper-16's current video memory format which can be one of *SNP16_FORMAT_MONO7, SNP16_FORMAT_MONO8* or *SNP16_FORMAT_YUV422*.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

The <format> is returned in the lower 16 bits, if successful.

There are no known bugs.

SEE ALSO

SNP16_set_format, *SNP16_get_decode_mode*.

SNP16_get_hue

USAGE

Terr SNP16_get_hue(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function returns the current hue control setting and is in the range 0x7F (+178.6º) to 0x80 (-180º).

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

The <hue> is returned in the lower 8 bits, if successful.

There are no known bugs.

SEE ALSO

SNP16_set_hue, SNP16_set_chrominance.

SNP16_get_ID

USAGE

Terr SNP16_get_ID(Thandle Hsnp16)

ARGUMENTS

Hsnp16 Handle to Snapper-16.

DESCRIPTION

This function returns the hardware identifier of the Snapper, so that an application can check whether it is running on a Snapper-16 or a Snapper-PCI16.

RETURNS

This function returns the hardware identifier - either *SNP16_ID* or *SNP16_PCI_ID*. Possible error codes:

ASLERR_BAD_HANDLE The Snapper-16 handle is invalid.

EXAMPLES

The following code checks whether a Snapper-16 is fitted to a module or it is a combined Snapper-PCI16 product.

```
if ( SNP16_get_ID(Hsnp16) == SNP16_ID )
   printf("Detected a Snapper-16 module");
else if ( SNP16_get_ID(Hsnp16) == SNP16_PCI_ID )
   printf("Detected a Combined Snapper-PCI16");
```

BUGS / NOTES

There are no known bugs.

This function is included for compatibility with existing applications. All new applications should use *SNP16_get_parameter*.

SEE ALSO

SNP16_get_parameter, *SNP16_get_rev*.

SNP16_get_parameter

USAGE

Terr SNP16_get_parameter(Thandle Hsnp16, ui16 parameter)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
Parameter	The parameter to return.

DESCRIPTION

This function returns various parameters from the internal structure associated with the Snapper-16 handle.

PARAMETER

SNP16_BASEBOARD_HANDLE	The handle to the baseboard which the Snapper-16 is fitted on. Type *(ui32)*.
SNP16_MAPPER_INTERFACE_TYPE	This is used as part of the initialisation code (see *SNP16_initialize*) to determine which Mapper type is fitted. After initialisation is complete, it is not required again. Type *(ui16)*.
SNP16_ID_VALUE	This returns the ID as read from the hardware, which distinguishes between all the different Snapper-16 variants. Type *(ui8)*.
SNP16_REV_VALUE	This returns the board revision as read from the hardware, which distinguishes between the different hardware revisions of the Snapper. Type *(ui8)*.
SNP16_IDREV_VALUE	This returns the ID and board revision as read from the hardware, which distinguishes between the different hardware revisions of the Snapper variants. Type *(ui8)*.
SNP16_FAMILY_VALUE	This returns *SNP16_FAMILY_ID*. Although this parameter always returns the same value, it is used to provide a mechanism for distinguishing future Snapper-16 variants, and for compatibility with the *SNP24_get_parameter* call. Type *(ui8)*.

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.
ASLERR_BAD_PARAM	The parameter value is invalid.
ASLERR_NOT_RECOGNIZED	The ID value read back from the Snapper hardware is not recognized.

EXAMPLES

The following example checks user input against the Snapper family to determine whether the hardware supports the request.

```
if (SNP16_get_parameter(Hsnp16, SNP16_FAMILY_VALUE) != SNP16_FAMILY_ID)
{
   /* Flag an error that this is a new Snapper-16 variant, which may require a
       software upgrade to support. */
}
```

BUGS / NOTES

The function returns a type *Terr* (*ui32* - an unsigned 32 bit integer). Therefore a cast may be need depending on the parameter type (given above for each parameter).

There are no known bugs.

SEE ALSO

SNP16_get_property.

SNP16_get_property

USAGE

*Terr SNP16_get_property(Thandle Hsnp16, char *property, char *value)*

ARGUMENTS

Hsnp16	Handle to Snapper-16.
Property	A character string or name of the property to access.
Value	The property result string. (Must point to a buffer of at least 16 bytes.)

DESCRIPTION

This function returns various property strings associated with Snapper-16.

PROPERTY

"fpgadate" **Snapper FPGA Date**: This retrieves the date and time string associated with current control FPGA file in use. It is unlikely that this function will ever be needed, but it can be useful to detect old versions of Snapper control FPGA information. (i.e. the date string is used as a revision level). The format of the returned date string is dd-mmm-yyyy hh:mm.

RETURNS

This function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.
ASLERR_BAD_PARAM	The property value is invalid.

EXAMPLES

To print the FPGA date:

```
char string[256];

SNP16_get_property(Hsnp16, "fpgadate", string);
printf("Snapper-16 FPGA date: %s", string);
```

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_get_parameter.

SNP16_get_rev

USAGE

Terr SNP16_get_rev(Thandle Hsnp16)

ARGUMENTS

Hsnp16 Handle to Snapper-16.

DESCRIPTION

This function returns the hardware revision level of the Snapper.

RETURNS

This function returns the hardware revision level. Possible error codes:

ASLERR_BAD_HANDLE The Snapper-16 handle is invalid.

EXAMPLES

```
if ( SNP16_get_rev(Hsnp16) == 0 )
   printf("\nRunning on issue 1 Snapper-16");
else if ( SNP16_get_rev(Hsnp16) == 1 )
   printf("\nRunning on issue 2 Snapper-16");
```

BUGS / NOTES

There are no known bugs.

This function is included for compatibility with existing applications. All new applications should use *SNP16_get_parameter*.

SEE ALSO

SNP16_get_parameter, *SNP16_get_ID*.

SNP16_get_ROI

USAGE

Terr SNP16_get_ROI(Thandle Hsnp16, i16 roi[ASL_SIZE_2D_ROI])

ARGUMENTS

Hsnp16 Handle to Snapper-16.

roi ROI array with four elements, with #defined element names:

ASL_ROI_X_START	Horizontal start position of ROI (0 = left of image).
ASL_ROI_Y_START	Vertical start position of ROI (0 = top of image).
ASL_ROI_X_LENGTH	Horizontal width of ROI.
ASL_ROI_Y_LENGTH	Vertical height of ROI.

The values in the *roi* array passed in are ignored.

DESCRIPTION

This function fetches the current ROI (Region of Interest) and returns it in the *roi* array.

The top left corner of the image is defined by the *ASL_ROI_X_START* and *ASL_ROI_Y_START* values and the image size defined with the *ASL_ROI_X_LENGTH* and *ASL_ROI_Y_LENGTH* values. All the coordinates are based upon raw image sizes in pixels and lines, ***not*** sub-sampled ones.

In line scan mode *ASL_ROI_Y_START* is always 0, and *ASL_ROI_Y_LENGTH* is the number of lines which get captured per bank.

RETURNS

This function returns the current ROI in the *roi* array. Possible error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

EXAMPLES

To display the current ROI:

```
SNP16_get_ROI(Hsnp16, roi);
printf("\nROI X start is %d", (int)roi[ASL_ROI_X_START]);
printf("\nROI Y start is %d", (int)roi[ASL_ROI_Y_START]);
printf("\nROI X length is %d", (int)roi[ASL_ROI_X_LENGTH]);
printf("\nROI Y length is %d", (int)roi[ASL_ROI_Y_LENGTH]);
```

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_ROI, *SNP16_get_ROI_max*.

SNP16_get_ROI_max

USAGE

Terr SNP16_get_ROI_max(Thandle Hsnp16, i16 roi[ASL_SIZE_2D_ROI])

ARGUMENTS

Hsnp16	Handle to Snapper-16.
roi	ROI array with four elements, with #defined element names:
	ASL_ROI_X_START Horizontal start position of ROI (0 = left of image).
	ASL_ROI_Y_START Vertical start position of ROI (0 = top of image).
	ASL_ROI_X_LENGTH Horizontal width of ROI.
	ASL_ROI_Y_LENGTH Vertical height of ROI.
	The values in the *roi* array passed in are ignored.

DESCRIPTION

This function fetches the maximum usable ROI (Region of Interest) for the camera in use and returns it in the *roi* array.

The maximum size is defined by the *ASL_ROI_X_LENGTH* and *ASL_ROI_Y_LENGTH* values, so the *ASL_ROI_X_START* and *ASL_ROI_Y_START* values are always returned as '0'. The coordinates are based upon raw image sizes in pixels and lines, ***not*** sub-sampled ones.

RETURNS

This function returns the maximum ROI in the *roi* array. Possible error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

EXAMPLES

To set the maximum allowable ROI:

```
SNP16_get_ROI_max(Hsnp16, roi);
SNP16_set_ROI(Hsnp16, roi[ASL_ROI_X_START], roi[ASL_ROI_Y_START],
       roi[ASL_ROI_X_LENGTH], roi[ASL_ROI_X_LENGTH]);
```

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_get_ROI, *SNP16_set_ROI*.

SNP16_get_subsample

USAGE

*Terr SNP16*_get_subsample(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function returns Snapper-16's current sub-sampling setting which can be one of *SNP16_SUB_X1, SNP16_SUB_X2, SNP16_SUB_X4* or *SNP16_SUB_X1_FIELD_DUPLICATE.*

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

The <sub_sample> is returned in the lower 8 bits, if successful.

There are no known bugs.

SEE ALSO

SNP16_set_subsample.

SNP16_initialize

USAGE

Tboolean SNP16_initialize(Thandle Hsnp16)

ARGUMENTS

Hsnp16 Handle to Snapper-16.

DESCRIPTION

This function is used to initialize the Snapper-16 module.

It makes calls to *SNP16_set_trigger_timeout*, *SNP16_set_ext_trigger*, *SNP16_set_decode_mode*, *SNP16_set_sync_start*, *SNP16_set_sync_stop*, *SNP16_set_clamp_start*, *SNP16_set_clamp_stop*, *SNP16_set_hue*, *SNP16_set_chrominance*, *SNP16_set_mono7_offset*, *SNP16_set_deinterlace*, *SNP16_set_filter*, *SNP16_set_format*, *SNP16_set_subsample*, *SNP16_set_input_mode*, *SNP16_set_video_source* and *SNP16_set_ROI*.

RETURNS

This function will either return *ASL_OK* or an error value from one of the lower level function calls listed above.

BUGS / NOTES

There are no known bugs.

SEE ALSO

-

SNP16_is_50Hz, SNP16_is_60Hz

USAGE

Tboolean SNP16_is_50Hz(Thandle Hsnp16)

Tboolean SNP16_is_60Hz(Thandle Hsnp16)

ARGUMENTS

Hsnp16 Handle to Snapper-16.

DESCRIPTION

These functions return *TRUE* or *FALSE* depending on the type of video signal present.

RETURNS

TRUE or *FALSE*.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_is_colour, *SNP16_is_locked*.

SNP16_is_capture_complete

USAGE

Tboolean SNP16_is_capture_complete(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function is used to test whether the current video capture has completed. It returns *FALSE* from the time that a capture is initiated with *SNP16_start_capture* until there is valid data in the video memory. *FALSE* is also returned after the capture has been initiated, but before a valid external trigger has occurred. *TRUE* is returned at all other times.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_start_capture, *SNP16_capture*, *SNP16_is_trigger_started*.

SNP16_is_colour

USAGE

Tboolean SNP16_is_colour(Thandle Hsnp16)

ARGUMENTS

Hsnp16 Handle to Snapper-16.

DESCRIPTION

This function returns *TRUE* if the video signal is colour, or *FALSE* if it is monochrome.

After switching the input signals (using *SNP16_set_video_source*) it may take up to a second to achieve colour lock, therefore there should be a small time delay before using this function if the inputs are switched.

RETURNS

TRUE or *FALSE*.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_is_locked, *SNP16_is_50Hz*, *SNP16_is_60Hz*.

SNP16_is_locked

USAGE

Tboolean SNP16_is_locked(Thandle Hsnp16)

ARGUMENTS

Hsnp16 Handle to Snapper-16.

DESCRIPTION

This function returns *TRUE* if Snapper-16 is locked to the video signal, or *FALSE* if it is not locked. This function can be used to determine whether or not there is a video signal present.

RETURNS

TRUE or *FALSE*.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_is_colour, *SNP16_is_50Hz*, *SNP16_is_60Hz*.

SNP16_is_trigger_started

USAGE

Tboolean SNP16_is_trigger_started(Thandle Hsnp16)

ARGUMENTS

Hsnp16 Handle to Snapper-16.

DESCRIPTION

This function is used to test whether the active edge of the external trigger input has occurred. It returns *TRUE* from the first active edge of the external trigger until the capture has completed. At all other times *FALSE* is returned.

RETURNS

TRUE or *FALSE*.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_is_capture_complete, *SNP16_set_ext_trigger*.

SNP16_read_video_data

USAGE

Terr SNP16_read_video_data(Thandle Hsnp16, Thandle Himage, ui16 TMG_action)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
Himage	Handle to an image.
TMG_action	*TMG_RUN* for normal operation, or *TMG_RESET* to abort.

DESCRIPTION

This function reads the data from Snapper-16 video memory into the image structure referenced by *Himage*. The data must already have been stored into video memory with a call to either *SNP16_capture* or *SNP16_start_capture*, and *Himage* must have been set up with a call to *SNP16_set_image*.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_deinterlace, *SNP16_reset_read_pointer*.

SNP16_reset_read_pointer

USAGE

Terr SNP16_reset_read_pointer(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function resets the read pointers to the video memory. This is used when an application requires to re-read the data in video memory. This routine will only function if a capture is not in progress.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_read_video_data.

SNP16_set_chrominance

USAGE

Terr SNP16_set_chrominance(Thandle Hsnp16, ui8 chrominance)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
chrominance	Chrominance level of the colour decoder.

DESCRIPTION

This function controls the gain of the chrominance signal. It is set to default values by the *SNP16_configure_xxx* functions. 0 is minimum gain and 255 is maximum gain.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

This function directly controls register 0x11 in the Philips SAA7191 decoder fitted to Snapper-16.

There are no known bugs.

SEE ALSO

SNP16_get_chrominance, *SNP16_set_hue*.

SNP16_set_clamp_start, SNP16_set_clamp_stop

USAGE

Terr SNP16_set_clamp_start(Thandle Hsnp16, ui8 clamp_start)

Terr SNP16_set_clamp_stop(Thandle Hsnp16, ui8 clamp_stop)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
Clamp_start	Start point of video clamp signal.
Clamp_stop	Stop point of video clamp signal.

DESCRIPTION

SNP16_set_clamp_start controls the start point of the clamp signal and is used to interface to non-standard video signals. The clamp start position is set to a default value at startup and should only be modified under special circumstances.

SNP16_set_clamp_stop controls the stop point of the clamp signal and is used to interface to non-standard video signals. The clamp stop position is set to a default value at startup and should only be modified under special circumstances.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

SNP16_set_clamp_start and *SNP16_set_clamp_stop* directly control registers 0x03, 0x16, and 0x04, 0x17 respectively in the Philips SAA7191 decoder fitted to Snapper-16.

This function directly controls registers 0x04 and 0x17 in the Philips SAA7191 decoder fitted to Snapper-16.

There are no known bugs.

SEE ALSO

SNP16_set_sync_start, SNP16_set_sync_stop.

SNP16_set_decode_mode

USAGE

Terr SNP16_set_decode_mode(Thandle Hsnp16, ui8 decode_mode)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
decode_mode	Colour decode mode.

DESCRIPTION

This function selects the colour decode mode which can take one of *SNP16_DECODE_PAL, SNP16_DECODE_NTSC* or *SNP16_DECODE_SECAM.* This function is called by *SNP16_configure_NTSC, SNP16_configure_PAL.*

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_get_decode_mode, SNP16_set_input_mode.

SNP16_set_deinterlace

USAGE

Terr SNP16_set_deinterlace(Thandle Hsnp16, Tboolean mode)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
mode	Deinterlace flag.

DESCRIPTION

This function controls how *SNP16_read_video_data* reads frames. When a frame of video data is captured the image can be deinterlaced automatically as it is read from video memory by setting *mode* to *TRUE* (the default as set on startup). If it required to preserve the fields as they were captured set *mode* to *FALSE*.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_read_video_data.

SNP16_set_ext_polarity

USAGE

*Terr SNP16_set_ext_polarity(Than*dle Hsnp16, Tboolean polarity)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
polarity	Polarity of the external trigger input.

DESCRIPTION

This function controls whether an external trigger capture is initiated on the rising or falling edge of the external trigger input. It accepts either *SNP16_RISING_EDGE* or *SNP16_FALLING_EDGE* as its input values.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_ext_trigger, *SNP16_set_trigger_timeout*.

SNP16_set_ext_trigger

USAGE

Terr SNP16_set_ext_trigger(Thandle Hsnp16, Tboolean trigger)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
trigger	External trigger control flag.

DESCRIPTION

This function controls whether a capture is triggered from the external trigger input.

If *trigger* is *TRUE* then image capture requested by *SNP16_capture* will not start until an active edge of the external trigger occurs. The active edge is selected via a call to *SNP16_set_ext_polarity*.

If *trigger* is *FALSE* then image capture requested by *SNP16_capture* will occur independent of the trigger status.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_ext_polarity, *SNP16_set_trigger_timeout*, *SNP16_is_trigger_started*.

SNP16_set_filter

USAGE

Terr SNP16_set_filter(Thandle Hsnp16, Tboolean filter)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
filter	Filter control flag.

DESCRIPTION

This function controls the horizontal filtering of the video data, and is used to reduce the affects of aliasing when the input image is sub-sampled. Note that only the luminance channel of the video data is filtered. This is considered acceptable as the colour channel generally does not contain high spatial frequencies.

The *filter* flag can take the Boolean values of *TRUE* for filtering enabled or *FALSE* for filtering disabled.

Setting sub-sampling to x2 or x4 (using *SNP16_set_subsample*) automatically enables this filter and setting to x1 automatically disables it.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

-

SNP16_set_format

USAGE

Terr SNP16_set_format(Thandle Hsnp16, Tparam snap_format, ui16 TMG_format)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
snap_format	Required format of Snapper.
TMG_format	Required TMG format of resulting Himage.

DESCRIPTION

This function controls the format in which data is stored in both Snapper-16 video memory and the host computer's memory. It is called by the *SNP16_configure_xxx* functions. The parameter *TMG_format* should be a TMG library pixel format (see the TMG Programmer's Manual).

SNAP_FORMAT

The parameter *snap_format* should be one of the following:

SNP16_FORMAT_MONO8	For standard 8 bit grayscale.
SNP16_FORMAT_MONO7	For 7 bit grayscale, with the MSB settable using *SNP16_set_mono7_offset*.
SNP16_FORMAT_YUV422	For colour 16 bit YUV 4:2:2 data stored in Snapper-16 video memory as YUYV, where YU is the first 16 bit word and YV the second

Many combinations of *snap_format* and *TMG_format* are only supported on baseboards with a data mapper, because it is the data mapper which performs the necessary format conversions. Even with a data mapper some combinations are not supported, for instance current data mappers do not support colour space conversion, so *TMG_RGB24* cannot be used as an output format in conjunction with *SNP16_FORMAT_YUV422*. When a combination is not supported this function returns an error, and a *TMG_image_convert* call should be made to perform the conversion in software.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_mono7_offset, *SNP16_set_input_mode*.

SNP16_set_hue

USAGE

Terr SNP16_set_hue(Thandle Hsnp16, ui8 hue)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
hue	Hue control.

DESCRIPTION

This function sets the hue of the data from the colour decoder, and is set to a default value by at startup. It can accept values in the range 0x7F (+178.6º) to 0x80 (-180º).

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

This function directly controls register 0x07 in the Philips SAA7191 decoder fitted to Snapper-16.

There are no known bugs.

SEE ALSO

SNP16_get_hue, SNP16_set_chrominance.

SNP16_set_image

USAGE

Terr SNP16_set_image(Thandle Hsnp16, Thandle Himage)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
Himage	Handle to an image.

DESCRIPTION

This function initializes the image structure into which the captured image will be stored. It sets the image based on the ROI (region of interest), the sub-sample ratio, and the image type based on the requested format. It also initializes the image to process the video data in one strip (see the TMG Programmer's Manual for an explanation of strip processing).

This function must be called after the width or height of the ROI, the sub-sample ratio, or the image format has been changed, but before *SNP16_read_video_data* is called.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

-

SNP16_set_input_mode

USAGE

Terr SNP16_set_input_mode(Thandle Hsnp16, ui8 input_mode)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
input_mode	Input mode selection.

DESCRIPTION

INPUT_MODE

This function selects the type of input signal and can take one of the following values:

SNP16_INPUT_COMPOSITE	Colour composite video input is selected. This mode is selected by: *SNP16_configure_NTSC*, *SNP16_configure_PAL*.
SNP16_INPUT_YC	Colour YC (S-Video) video input is selected.
SNP16_INPUT_MONO	Monochrome video input is selected. This mode is selected by: *SNP16_configure_gray_60Hz*, *SNP16_configure_gray_50Hz*.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_video_source, *SNP16_set_format*, *SNP16_set_decode_mode*.

SNP16_set_mode_vcr

USAGE

Terr SNP16_set_mode_vcr(Thandle Hsnp16, Tboolean mode)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
mode	VCR mode flag - either *TRUE* or *FALSE*.

DESCRIPTION

This function sets the acquisition hardware's phase locked loop (PLL) to allow locking to a video cassette recorder (VCR) signal. A video signal from a VCR is generally not very stable and therefore requires the PLL to have a different response.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

This function directly controls the VTRC bit in register 0x0D in the Philips SAA7191 decoder fitted to Snapper-16.

There are no known bugs.

SEE ALSO

-

SNP16_set_mono7_offset

USAGE

Terr SNP16_set_mono7_offset(Thandle Hsnp16, Tboolean offset)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
offset	MSB in 7 bit mono mode - either *TRUE* or *FALSE*.

DESCRIPTION

This function controls the MSB when Snapper-16 is in 7 bit mono mode. This mode is potentially useful because only 128 gray levels are used in 7 bit mode. 256 colour displays can therefore still display other colours as well as a grayscale image directly from Snapper-16.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_format.

SNP16_set_ROI

USAGE

Terr SNP16_set_ROI(Thandle Hsnp16, Tcoord x_start, Tcoord y_start, Tcoord width, Tcoord height)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
x_start	Horizontal start position of ROI (0 = left of image).
y_start	Vertical start position of ROI (0 = top of image).
width	Horizontal width of ROI.
height	Vertical height of ROI.

DESCRIPTION

This function defines the ROI (region of interest). It accepts an X,Y start coordinate (0,0 is top left) and the region width and height. The function is called with default values by the *SNP16_configure_xxx* functions.

All the coordinates are based upon raw image sizes in pixels and lines, **not** sub-sampled ones. This allows the image sub-sampling ratio to be varied for fast update or image resolution, without varying the ROI as well.

The horizontal and vertical resolutions are 8 pixels per line and 2 lines per field respectively. If the coordinates are not correctly aligned the values are rounded down to the nearest 8 pixel or 2 line multiple.

The values are checked against maximum screen sizes depending whether 50Hz or 60Hz video is being acquired. If any point within the ROI falls outside the active area for the currently selected video standard the routine will trim the ROI so that it is within the active area. This is done to simplify the use of the function with interactive software (e.g. window sizing).

The benefit of using a reduced ROI compared to a full screen image is that the frame readout rate can be significantly faster because there is less data to read out.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

When a new ROI has been set there is a single field latency before the hardware processes the new values.

SEE ALSO

SNP16_get_ROI, SNP16_get_ROI_max.

SNP16_set_subsample

USAGE

Terr SNP16_set_subsample(Thandle Hsnp16, ui16 subsample)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
subsample	Sub-sample ratio.

DESCRIPTION

This function controls sub-sampling the image.

SUBSAMPLE

Subsample should be one of the following:

SNP16_SUB_X1	This selects full resolution image capture. Horizontal filtering is disabled.
SNP16_SUB_X2	This selects sub-sampled by 2 image capture. This is achieved by capturing every other pixel in the horizontal direction and only the first field in the vertical direction. Horizontal filtering is enabled
SNP16_SUB_X4	This selects sub-sampled by 4 image capture. This is achieved by capturing every fourth pixel in the horizontal direction and every other line of the first field in the vertical direction. Horizontal filtering is enabled.
SNP16_SUB_X1_FIELD_DUPLICATE	This selects full width image capture, but only one field which is then duplicated to make a complete frame. This halves the vertical resolution, but allows fast movement to be captured without the 'double image' which would result if movement was captured in normal full frame mode from a field exposure camera. Horizontal filtering is disabled. If Snapper-16 is used in a system with a slow bus compared to the processor speed (e.g. most ISA systems) then this mode will allow faster transfer rates than *SNP16_SUB_X1*. Conversely, in a system with a fast bus which supports DMA (e.g. PCI or SBus systems) this mode will be slower than *SNP16_SUB_X1*.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.
ASLERR_BAD_PARAMETER	The sub sample mode is invalid.

BUGS / NOTES

If the application starts up in x2 or x4 sub-sampling mode and then switches to x1 sub-sampling, one field of the next captured frame is corrupted. However this only occurs once and subsequent switches between sub-sampling modes function correctly.

SNP16_SUB_X1_FIELD_DUPLICATE used to be called *SNP16_SUB_FAST*. When JPEG compressing directly from a Snapper-16 using a Crunch board, the *SNP16_SUB_X1_FIELD_DUPLICATE* option actually only compresses a single field (rather than duplicating it).

SEE ALSO

SNP16_set_filter.

SNP16_set_sync_start, SNP16_set_sync_stop

USAGE

Terr SNP16_set_sync_start(Thandle Hsnp16, ui8 sync_start)

Terr SNP16_set_sync_stop(Thandle Hsnp16, ui8 sync_stop)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
sync_start	Start position of sync signal.
sync_stop	Stop position of sync signal.

DESCRIPTION

SNP16_set_sync_start controls the start point of the sync signal and is used to interface to non-standard video signals. The sync start position is set to a default value at startup and should only be modified under special circumstances.

SNP16_set_sync_stop controls the stop point of the sync signal and is used to interface to non-standard video signals. The sync stop position is set to a default value at startup and should only be modified under special circumstances.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

These functions directly control registers 0x01, 0x14, 0x02 and 0x15 in the Philips SAA7191 decoder fitted to Snapper-16.

There are no known bugs.

SEE ALSO

SNP16_set_clamp_start, SNP16_set_clamp_stop.

SNP16_set_trigger_timeout

USAGE

Terr SNP16_set_trigger_timeout(Thandle Hsnp16, ui32 timeout)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
timeout	Timeout value in milliseconds.

DESCRIPTION

This function sets the time that *SNP16_capture* or *SNP16_is_capture_complete* will wait for a trigger pulse before it returns *ASLERR_TIMEOUT*.

It can be called with the following #defines: *MILLISECONDS, SECONDS*, or *MINUTES*. For example:

```
SNP16_set_trigger_timeout(Hsnp16,  2 MINUTES);
```

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_ext_trigger, *SNP16_capture*.

SNP16_set_vertical_mode

USAGE

Terr SNP16_set_vertical_mode(Thandle Hsnp16, ui8 mode)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
mode	Vertical timing mode.

DESCRIPTION

This function sets Snapper-16's vertical timing mode.

MODE

There are four modes:

SNP16_VERT_NORMAL	This mode should be used with standard cameras, and it is set as default at startup.
SNP16_VERT_SEARCH	This mode should be used to achieve fastest vertical locking following a change of video input. Note: For fastest locking select VCR mode as well by using *SNP16_set_mode_vcr*.
SNP16_VERT_BYPASS	This mode should be used with asynchronous reset cameras.
SNP16_VERT_AUTO	This mode selects the "auto-deflection" mode of the Philips SAA7191 decoder fitted to Snapper-16.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

This function directly controls the VN01 bits in register 0x10 in the Philips SAA7191 decoder fitted to Snapper-16.

There are no known bugs.

SEE ALSO

-

SNP16_set_video_source

USAGE

Terr SNP16_set_video_source(Thandle Hsnp16, ui8 video_source)

ARGUMENTS

Hsnp16	Handle to Snapper-16.
video_source	Video source input channel.

DESCRIPTION

This function sets Snapper-16's video source channel. It accepts a video source channel number 1, 2 or 3. This function is used to multiplex up to three video sources into Snapper-16.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_set_input_mode.

SNP16_start_capture

USAGE

Terr SNP16_start_capture(Thandle Hsnp16)

ARGUMENTS

Hsnp16	Handle to Snapper-16.

DESCRIPTION

This function digitizes the next field or frame of video data into Snapper-16's on-board video memory. Snapper-16 must be initialized with the required video source, region of interest etc, prior to calling this function. This function starts acquisition and returns immediately. The controlling program must call *SNP16_is_capture_complete* to determine when the capture has completed. This mode allows the time taken by the hardware to store the video data to be used by the software to perform other processing. This method is useful in multitasking systems, or when the capture and display rate must be optimized.

RETURNS

The function returns the following error codes:

ASL_OK	If successful.
ASLERR_BAD_HANDLE	The Snapper-16 handle is invalid.

BUGS / NOTES

There are no known bugs.

SEE ALSO

SNP16_capture, *SNP16_is_capture_complete*.