SNAPPER Error Handling

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-ERR-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 error handling procedures used in the Snapper libraries.

Error Type Terr

Most Snapper library functions return a type *Terr*. *Terr* is a 32 bit unsigned integer, with the bit positions defined as follows:

31 to 24 Hardware identifier/revision or software module code, or 0 if no error.
This is used to allow a top level calling function to determine the library in which the error occurred.

23 to 16 Error code, otherwise 0 if no error.
See the Error Codes section, on page 3, for further details.

15 to 0 Function return value.
This is used to return valid numbers if no error occurs, or to provide additional information in the event of an error.

Error Codes

The following is a list of error codes used by all Snapper libraries, and a description of each error. The error codes are generic, as a result the codes may have a wider usage in future than the descriptions currently imply.

BAD_XXX ERRORS

ASLERR_BAD_HANDLE	The handle has not been set up, or is corrupt.
ASLERR_BAD_REG	An attempt has been made to access a non-existent hardware register.
ASLERR_BAD_IMAGE	The image structure has not been set up, or is corrupt.

'NOT POSSIBLE' ERRORS

ASLERR_NOT_SUPPORTED	The requested operation is not supported, and is unlikely to be supported in future.
ASLERR_NOT_IMPLEMENTED	The requested operation is not currently implemented, but may be implemented in future.
ASLERR_INCOMPATIBLE	The requested option is not compatible with existing Snapper settings.
ASLERR_NOT_RECOGNIZED	The requested option is not recognized.

FUNCTION PARAMETER ERRORS

ASLERR_BAD_PARAM	A parameter passed to a function has not been recognised.
ASLERR_OUT_OF_RANGE	A parameter passed to a function is invalid - typically too large or too small.
ASLERR_PARAM_CONFLICT	Two or more parameters passed to a function are mutually exclusive.

HARDWARE RELATED ERRORS

ASLERR_HW_FAILURE	A problem has occurred which may indicate a hardware problem. Check that the Snapper is installed correctly and is not clashing with other boards in the computer.
ASLERR_HW_NOT_FOUND	The requested Snapper or baseboard has not been detected.
ASLERR_UNKNOWN_BASEBOARD	The baseboard has not been recognised.
ASLERR_UNKNOWN_MODULE	The Snapper module has not been recognised. Check that a module is fitted to the selected baseboard. This error can also occur if the _SNPxx compiler directive is missing (see the ***Snapper! Developer's Guide***).
ASLERR_OUT_OF_LOCK	This is typically used when a Snapper has failed to lock to incoming video.
ASLERR_NO_VIDEO	No video source has been found on the selected input.
ASLERR_TIMEOUT	An operation took longer than expected.
ASLERR_OVERFLOW	Some form of internal overflow has occurred.
ASLERR_UNDERFLOW	Some form of internal underflow has occurred.
ASLERR_PARITY	A parity error has occurred (typically on serial communications).

OPERATING SYSTEM ERRORS

ASLERR_OUT_OF_MEMORY	A system call to reserve memory has failed.
ASLERR_THREAD_ERROR	A system call to control a separate thread of execution has failed.
ASLERR_DRIVER_CALL_FAILED	A call to the Snapper device driver has failed. For operating systems with a console (e.g. Solaris) check the console for any error messages from the driver.
ASLERR_SYSTEM_CALL_FAILED	An operating system call (other than those listed above) failed. For MS-DOS and Windows 3.1 this is used for BIOS and graphics driver calls.

FILE AND RELATED ERRORS

ASLERR_OPEN_FAILED	Open failed.
ASLERR_CLOSE_FAILED	Close failed.
ASLERR_READ_FAILED	Read failed.
ASLERR_WRITE_FAILED	Write failed.
ASLERR_SEEK_FAILED	Seek operation failed.
ASLERR_CORRUPT	File or data stream is corrupt.

MISCELLANEOUS ERRORS

ASLERR_OUT_OF_HANDLES	No free handles were found.
ASLERR_INTERNAL_ERR	An internal error was detected in the libraries.
ASLERR_IN_PROGRESS	The requested operation is already in progress.
ASLERR_INVALID_STATE	The library has detected an invalid state in the software or hardware, but cannot determine a more specific cause of the problem.
ASLERR_INCOMPATIBLE_LIBRARIES	One or more library files (DLLs, ".so" etc) installed on the system have incompatible revisions. This error is also returned if the device driver installed is incompatible with the libraries. If this error occurs check if multiple copies of library files are installed on the computer. If in doubt as to which files are correct, delete all old library files and re-install the software.

WARNINGS

ASLWRN_TIMEOUT	Although an operation took longer than expected, and caused the function call to fail, the error handler will not be called. This is mainly used in serial communications where timeout failures may be required as part of a protocol implementation, for example in Xmodem.

Function Overview

The macros *ASL_is_err* and *ASL_is_ok* allow the *Terr* return value from a function to be tested to see if an error has occurred. Similarly, *ASL_get_err* and *ASL_get_ret* allow the error code to be separated from the main value returned.

The macro *ASL_roe* is a simple way of 'rippling back' an error to a higher level function if the calling function does not want to deal with the error.

If an error occurs in the library an error message can be displayed, typically in a windowed environment as a popup message box. The function *ASL_err_set_reporting* controls whether or not a message is displayed, and if it is displayed, whether the standard message display routine *ASL_err_display* is used, or a user defined one.

By default no messages are displayed, so it is recommended at least for initial debugging of an application, that *ASL_err_set_reporting* is called at the start of the application to install *ASL_err_display*. A finished application may make many calls to *ASL_err_set_reporting* to allow some error conditions to be reported to the user, but others to be handled quietly within the application.

Function List

ERROR FUNCTIONS

ERROR MACROS

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

ASL_err_display

USAGE

*void ASL_err_display(char *Pfunc_name, Terr error, char *Pdescription)*

ARGUMENTS

Pfunc_name	Name of the function which generated the error.
error	Error code.
Pdescription	Additional text information supplied by the function which caused the error.

DESCRIPTION

This is the default error display handler which is called on an error unless an alternative handler has been installed. It should not be called directly. It will generate error displays of the style:

Pfunc_name: Failed with error code *error*
(Generic description of error)
Pdescription

for instance:

SNP24_capture: Failed with error code 0xB0410000
(Hardware timed out)
While waiting for trigger to be received

This will be displayed in a message box under a windows environments (e.g. Microsoft Windows 3.1/95/98/NT), or to *stdout* or *stderr* under MS-DOS and other operating systems.

EXAMPLES

The following code will restore *ASL_err_display* as the error handler:

```
ASL_err_set_reporting(ASL_ERR_SET_HANDLER, ASL_err_display);
```

BUGS / NOTES

There are no known bugs.

SEE ALSO

ASL_err_set_reporting.

ASL_err_set_reporting

USAGE

*Terr ASL_err_set_reporting(int mode, void (EXPORT_FN *handler)(char*, Terr, char*))*

ARGUMENTS

mode	Required mode.
handler	The error display handler to install.

DESCRIPTION

This function can be used to install a new error display handler for a specific application, to restore the built-in handler, or to turn off error reporting so that errors can be trapped internally and do not get displayed to the user. It should be called at the very start of an application to enable the required handler.

mode should be *ASL_ERR_SET_HANDLER*. No other options are currently supported.

Installing a new error display handler: *handler* should be a pointer to the application specific handler function as shown in the example below.

Restoring the built-in handler: *handler* should be a pointer to *ASL_err_display*.

Turning off error reporting: *handler* should be NULL.

RETURNS

Possible error codes:

ASL_OK	If successful.
ASLERR_BAD_PARAM	*mode* is invalid.

Note that if this function generates an error itself, the error display will be handled by whichever handler was installed before the function was called.

EXAMPLES

This example shows how the application would install a new error display handler, called *MyErrorDisplayHandler,* switch off error reporting, and finally restore the built-in error handler:

```
/* Install a new error display handler */
ASL_err_set_reporting(ASL_ERR_SET_HANDLER, MyErrorDisplayHandler);
...
/* Install a NULL handler - i.e. no error reporting */
ASL_err_set_reporting(ASL_ERR_SET_HANDLER, NULL);
...
/* Install the built-in error display handler */
ASL_err_set_reporting(ASL_ERR_SET_HANDLER, ASL_err_display);
```

This example shows the source code for *MyErrorDisplayHandler* used above. The example here is for Windows 3.1 but is basically the same for all operating systems:

```
// An example of how to use a custom error display handler
void EXPORT_FN MyErrorDisplayHandler(char *Pfunc_name, Terr error, char
      *Pdescription)
{
   char error_string[MAX_ERROR_STRING_LENGTH];

   // Build a string with the error message - first the calling function
   // and error code.
   wsprintf(error_string, "\n%s:  Failed with error code 0x%08lx", Pfunc_name,
       (long) error);

   // Add the description field, if present
   if (strncmp(Pdescription, "", MAX_ERROR_STRING_LENGTH) != 0) {
      strncat(error_string, "\n\n", MAX_ERROR_STRING_LENGTH);
      strncat(error_string, Pdescription, MAX_ERROR_STRING_LENGTH);
   }

   // Finally display the error as a messagebox.
   ::MessageBox(0, error_string, "MY ERROR HANDLER", MB_OK);
}
```

BUGS / NOTES

Error reporting is turned off by default, hence the recommendation to call this function at the start of every application.

SEE ALSO

ASL_err_display.

ASL_get_err, ASL_get_ret

USAGE

ui16 ASL_get_err(Terr return_value)
ui16 ASL_get_ret(Terr return_value)

ARGUMENTS

return_value Standard error return (32 bit unsigned integer).

DESCRIPTION

These macros extract parts of a Terr return value.

ASL_get_err extracts the error code value (i.e. bits D23 .. D16).

ASL_get_ret extracts the return value (i.e. bits D15 .. D00). (It does it by simply casting the *return_value* to a *ui16*.).

RETURNS

Requested part of *return_value* - see above.

EXAMPLES

```
if (ASL_is_ok(return_value = DIG16_get_subsample(Hdig16))
   if (ASL_get_ret(return_value) == DIG16_SUB_X2)
      printf("x2 sub-sampling");
else if (ASL_get_err(return_value) == ASLERR_BAD_HANDLE)
      printf("Snapper handle invalid");
```

BUGS / NOTES

There are no known bugs.

SEE ALSO

ASL_is_err, ASL_is_ok.

ASL_is_err, ASL_is_ok

USAGE

Tboolean ASL_is_err(Terr return_value)
Tboolean ASL_is_ok(Terr return_value)

ARGUMENTS

return_value Standard error return.

DESCRIPTION

These macros return a boolean depending on whether or not *return_value* indicates an error. (See "ERROR TYPES" on page 2).

ASL_is_err returns *TRUE* if *return_value* indicates an error, or *FALSE* if no error.

ASL_is_ok returns *TRUE* if *return_value* indicates no error, or *FALSE* if this is an error.

RETURNS

TRUE / *FALSE* - see above.

EXAMPLES

```
if (ASL_is_err(type = DIG16_get_camera_type(Hdig16)))
   printf("Failed to get camera type");
```

BUGS / NOTES

There are no known bugs.

SEE ALSO

ASL_get_err, ASL_get_ret.

ASL_roe

USAGE

void ASL_roe(Terr return_value)

ARGUMENTS

return_value Standard error return.

DESCRIPTION

This macro is the standard way of 'rippling back' an error return if the calling function does not want to deal with the error. "roe" stands for "Return On Error". *return_value* would normally be the *Terr* returned by a function called within the *ASL_roe* macro.

```
ASL_roe( function(...) );
```

is equivalent to:

```
status = function(...);
if (ASL_is_err(status))
   return(status);
```

Note that *ASL_roe* should not be used where some tidying up is needed before returning, for example:

```
buf_ptr = malloc(...);
...
status = function(...);
if (ASL_is_err(status))
{
   free(buf_ptr);
   return(status);
}
```

RETURNS

-

EXAMPLES

See above.

BUGS / NOTES

There are no known bugs.

SEE ALSO

ASL_is_err, *ASL_is_ok*, *ASL_get_err*, *ASL_get_ret*.