Multi-Purpose ASIC
Control & Interface Electronics
(MACIE)

Programmer's Guide

Version 5.2

June 24, 2021

Markury Scientific, Inc.

Table of Contents

1 Introduction

The MACIE library is a 64-bit software programming interface for the MACIE card. It provides the format of shared library (.so) for the Linux platform and dynamic link library (.dll) for the MS Windows platform. The MACIE library implements proprietary function protocols for the application software to control the MACIE card according to the diagram shown in figure 1. This document provides a description of the available API functions. In addition, some sample C code is provided as part of the distribution that illustrates the usage of these functions. While the library itself is written in C++ using the QT framework, the library interface is restricted to using only C-style declarations and functions to simplify the integration into the various available programming languages.

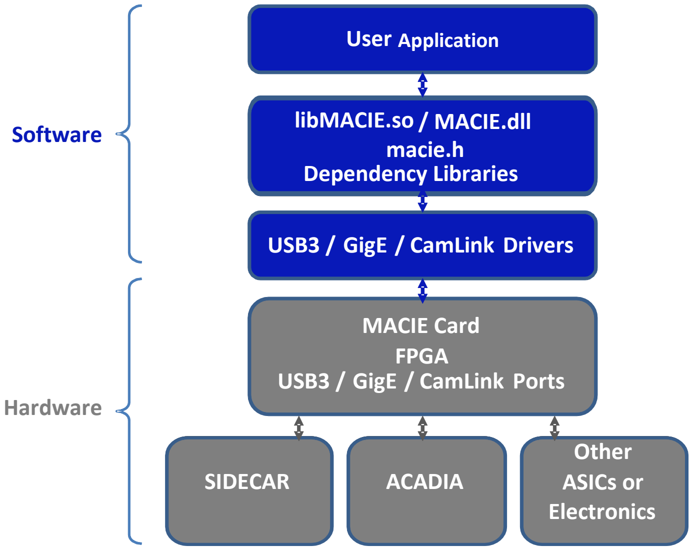

Figure 1 MACIE Library Architecture

The MACIE library implements 2 groups of API functions with 3 kinds of communication interfaces – USB, CamLink Serial Port, and GigE:

- Configuration functions for MACIE card and connected ASIC (SIDECAR, ACADIA and others), for example: reading / writing register, downloading firmware, image ramp configuration, telemetry
- Image acquisition functions

The MACIE library contains the following files:

- libMACIE.so for Linux and MACIE.dll for windows
- macie.h
- All the dependency libraries including the Qt libraries which are in the same directory of libMACIE.so / MACIE.dll

2 Software Installation and PC Configuration

The MACIE library is installed as part of the MACIE software package. Details about the package contents, the installation process, and the PC configuration can be found in the **MACIE Software Manual**. Please refer to sections 3 - 6, and section 8 of the software manual.

After installation, the required library files can be found in the ./bin (Windows) or MacieApp (Linux) folder of the installation directory, and include the following files:

• macie.h	- general C-style header file with all function and structure/enum definitions
• MACIE.dll	- Windows shared library (only included in Windows distribution)
• libMACIE.so	- Linux shared library (only included in Linux distribution)

MACIE Library Linkage: There are two options to link the MACIE library to a custom user program or development environment:

- Directly link the MACIE library in the installation path: bin\ (Windows) or MacieApp/ (Linux)
- Copy the entire bin or MacieApp folder to the user specified location. It is important to copy all the dependency libraries with the MACIE library together. The MAC and MSAC executable and the two corresponding ini files (mac.ini and msac.ini) can be ignored).

3 MACIE Functions

3.1 MACIE_LibVersion

```
float MACIE_LibVersion()
```

Summary

This function returns the version number of the MACIE library. This function can be called at any time, even before MACIE_Init is called.

Parameters

None

Return Value

Version number as floating point value, e.g. 2.1

3.2 MACIE_Init

```
MACIE_STATUS MACIE_Init()
```

Summary

This is the starting point for the application to interface with the MACIE library. This function does not power up or initialize the MACIE hardware system. However, it must be called prior to using any other API function to communicate with the MACIE system, as it configured and prepares internal variables and structures that are needed by the other functions.

To release any memory that is allocated by the MACIE_Init() function (e.g. before closing the user application), call the MACIE_Free() function.

Parameters

None

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

3.3 MACIE_Free

```
MACIE_STATUS MACIE_Free()
```

Summary

Free up all allocated resources including memory allocated with MACIE_Init().

Parameters

None

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

Comments

This function should be called at the end of an application to free up all allocated resources.

3.4 MACIE_Error

```
char* MACIE_Error()
```

Summary

Returns a text description of the last device error that occurred. When a MACIE function returns MACIE_FAIL, call this function for details regarding the type of failure.

Parameters

None

Return Value

A char text string with the error description.

3.5 MACIE_CheckInterfaces

```
MACIE_STATUS MACIE_CheckInterfaces( unsigned short gigeCommandPort,
                                    MACIE_IpAddr *pIpAddrList,
                                    unsigned short nIpAddr,
                                    unsigned short *numCards,
                                   MACIE_CardInfo **pCardInfo )
```

Summary

Find all available MACIE cards that are connected to the computer (directly or via network) and provide information about the available interfaces (Camera Link, GigE and USB) to each card.

Parameters

gigeCommandPort — Unsigned integer used for GigE port number. If 0, the default value 42306 will be used; otherwise the input value will be used. This port number has to match the port number configured in the MACIE Webserver for the TCP auto-connect selection (set to 42306 by default). – Input

pIpAddrList — Pointer to an array of MACIE_IpAddr to specify a list of GigE IP addresses to search. Use this parameter to provide a list of IP addresses to be included in the search for MACIE cards. This is only needed if the network prevents the automatic detection of MACIE cards. Use NULL if no IP address needs to be specified. - Input

nIpAddr — Number of GigE IP addresses included in the pIpAddrList above. Use 0 if no IP address is specified - Input

numCards — Pointer to an unsigned integer to store the number of MACIE cards connected. - Output.

pCardInfo — Pointer to an array of MACIE_CardInfo structures. - Output

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_Init() has been called.

The input parameter gigeCommandPort is useful if the user needs to modify the default command port number (42306) on the MACIE card. This could happen if the network blocks the default command port number. To change the port number, use the MACIE web server by calling the MACIE IP address from a browser, and then navigate to the Connection tab to change the port number in channel 0 of the Auto-Open Settings. After that, the new port number has to be used in the MACIE_CheckInterfaces function to communicate with the MACIE card.
To use the default port number, simply set the gigeCommandPort parameter to 0.

This function applies the default timeout of 200 ms for checking GigE communication interface. In order to apply a different timeout, please call MACIE_SetGigETimeout before calling this function.

The Array of `MACIE_CardInfo` contains all available interfaces on each card. The storage for the list is allocated by the library and is de-allocated by MACIE_Free() function.

The value field of the `MACIE_CardInfo` structure can be used to determine what interfaces are connected on each MACIE card.

Note: Any hardware configuration change like connecting additional MACIE cards or disconnecting an interface type (e.g. USB) from an existing MACIE card will not be detected until the MACIE_CheckInterfaces() function is called again.

3.6 MACIE_ SetGigeTimeout

```
unsigned long MACIE_SetGigeTimeout( unsigned short timeout)
```

Summary

Set a timeout for checking the GigE communication interface to any MACIE card on the network (when calling the MACIE_CheckInterfaces function). The default timeout of 200 ms will be applied if this function is not called. This timeout can be increased if the network is slow and the available MACIE cards cannot be reliably detected within the default timeout period.

Parameters

timeout	Unsigned 16 bit integer indicating the timeout in milliseconds for detecting GigE interface - Input.

Return Value

MACIE_OK if successful,

Comments

This function should be called before calling MACIE_CheckInterfaces()

3.7 MACIE_GetHandle

```
unsigned long MACIE_GetHandle( unsigned short MACIESerialNumber,
                               MACIE_Connection connection)
```

Summary

Set current communication interface with the input MACIE serial number and connection type. Then return a unique handle based on the input MACIE serial number and the MACIE_Connection type.

Parameters

MACIESerialNumber	Unsigned 16 bit integer indicating the MACIE serial number. - Input.
connection	enum value of MACIE_Connection. - Input

Return Value

A 32-bit unsigned integer. None-zero indicates successful; zero means invalid MACIE serial number or connection.

Comments

This function can only be called after MACIE_Init() has been called.

The input parameters of MACIE serial number and connection can be obtained from the MACIE_CardInfo list after calling MACIE_CheckInterfaces().

3.8 MACIE_GetAvailableMACIEs

```
unsigned char MACIE_GetAvailableMACIEs( unsigned long handle )
```

Summary

This function will report how many MACIE cards are connected through the same interface that is specified by the handle. Up to 8 MACIE cards can be plugged on top of each other through the board-to-board connectors, and can be accessed through a single interface to the computer.

Parameters

handle	Unsigned 32-bit integer obtained by MACIE_GetHandle() function. - Input.

Return Value

A 8-bit unsigned integer with each bit indicating a MACIE card (up to 8 cards total). If 0 is returned, no MACIE card is available or the MACIE check failed. For example:
0x03: means MACIE 0 and MACIE 1 are available.

Comments

This function has been provided for future use but can already be used to verify the availability of a given MACIE card. Multiple MACIE cards connected through the same interface (same handle) are not yet supported. Therefore, for now this function should always return 1 to indicate that exactly one MACIE card is connected with ID 0.

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.9 MACIE_GetAvailableASICs

```
unsigned char MACIE_GetAvailableASICs( unsigned long handle,
                                       int asicType )
```

Summary

This function will report how many ASICs are connected through the same interface that is specified by the handle. Up to 8 ASICs can be connected, using up to 8 separate MACIE cards plugged on top of each other using the board-to-board connectors, and can be accessed through a single interface to the computer.

Parameters

handle	Unsigned 32-bit integer obtained by MACIE_GetHandle() function. - Input.
asicType	reserved for future use

Return Value

A 8-bit unsigned integer with each bit indicating an ASIC (up to 8 ASICs total). If 0 is returned, no ASIC is available or the ASIC check failed.
For example: 0x03: means ASIC 0 and ASIC 1 are available.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

It also requires that the MACIE and ASIC are properly configured to communicate with each other (all drivers are on, clock rates are set, etc.). This usually requires downloading the MACIE register and possibly ASIC register files beforehand.

After checking available ASICs, this function also resets the error counters corresponding to the red-flashing LEDs for all the ASICs. The reset removes any errors detected due to non-connected ASICs (which are expected).

3.10 MACIE_ReadMACIEReg

```
MACIE_STATUS MACIE_ReadMACIEReg( unsigned long  handle,
                                 unsigned char  slctMACIEs,
                                 unsigned short address,
                                 unsigned int   *value  )
```

Summary

Read a MACIE register value.

Parameters

handle Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

address 16-bit unsigned integer indicating a MACIE register address. – Input

value Pointer to an unsigned integer indicating the value read from the register. - Output

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.11 MACIE_WriteMACIEReg

```
MACIE_STATUS MACIE_WriteMACIEReg( unsigned long  handle,
                                  unsigned char  slctMACIEs,
                                  unsigned short address,
                                  unsigned int   value )
```

Summary

Write value to MACIE register.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

address — 16-bit unsigned integer indicating a MACIE register address. – Input

value — Unsigned integer indicating the value to be written to the register. – Intput

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.12 MACIE_WriteMACIEBlock

```
MACIE_STATUS MACIE_WriteMACIEBlock( unsigned long  handle,
                                    unsigned char  slctMACIEs,
                                    unsigned short address,
                                    unsigned int   *valueArray,
                                    int            arrSize )
```

Summary

Write a block of values to a contiguous set of MACIE registers.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

address — 16-bit unsigned integer indicating the starting register address. – Input

valueArray — Pointer to an unsigned integer array. The values in the array will be written to the MACIE registers starting at address - Input

arrSize — Array size indicating how many registers will be written.

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

Example:

```
unsigned int arr = {1000, 2047, 15000};
long handle = 0x00010002;  // returned from MACIE_GetHandle()
MACIE_STATUS status = MACIE_WriteMACIEBlock(handle, 1, 0x01c1, arr, 3);
```

This will write 1000 to address 0x01c1, 2047 to address 0x01c2, and 15000 to address 0x01c3.

3.13 MACIE_ReadMACIEBlock

```
MACIE_STATUS MACIE_ReadMACIEBlock( unsigned long  handle,
                                   unsigned char  slctMACIEs,
                                   unsigned short address,
                                   unsigned int   *valueArray,
                                   int            arrSize )
```

Summary

Read a number of registers staring at the specified register address.

Parameters

handle Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

address 16-bit unsigned integer indicating the starting register address. – Input

valueArray Pointer or address of an unsigned integer array storing the readback register values. - Output

arrSize Array size indicating how many registers will be read. -Input

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

The memory of valueArray has to be allocated by application prior to calling this function.

Example:

```
uint *pData = new uint[3];
long handle = 0x00010002;  // returned from MACIE_GetHandle();
MACIE_STATUS status = MACIE_ReadMACIEBlock(handle, 1, 0x01c1,  pData, 3);
```

This will read from address 0x01c1, 0x01c2, and 0x01c3 and return the read values in pData.

3.14 MACIE_LoadMACIEFirmware

```
MACIE_STATUS MACIE_LoadMACIEFirmware( unsigned long handle,
                                      unsigned char slctMACIEs,
                                      bool          bSlot1,
                                      unsigned int  *pResult);
```

Summary

Each MACIE card stores two FPGA firmware versions in the available on-board EEPROM slots (slot1 and slot2). Calling this function will load the firmware from the specified slot.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

bSlot1 — Boolean value If true, load FPGA firmware from slot 1, otherwise load FPGA firmware from slot 2. – Input

pResult — Pointer to an integer indicating the value read from MACIE register 0xFFFB - Output

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

Loading the firmware is usually completed within 2 seconds. However, if communicating to the MACIE card through GigE, it may take up to 10 seconds until the GigE has been fully re-initialized, and the MACIE_LoadMACIEFirmware() function returns (first generation MACIE cards only, newer cards do not require re-initialization).

After power-cycling the MACIE card, this function should be called to load the desired firmware. Afterwards, re-loading the firmware is optional (e.g. when re-initializing the full system), but may be useful since it provides a clean slate of default MACIE register values.

It is recommended to verify the pResult to ensure that the correct FPGA firmware has been loaded. The following values are expected:

base mode firmware:	0xBCDE
SIDECAR firmware:	0xAC1E
ACADIA firmware:	0xACDA

In addition, the following MACIE register addresses can be read using the MACIE_ReadMACIEReg() or MACIE_ReadMACIEBlock() functions to obtain additional information about the loaded firmware:

0xfffd: Firmware version number
0xfffe: Firmware release month and day
0xffff: Firmware release year

3.15 MACIE_DownloadMACIEFile

```
MACIE_STATUS MACIE_DownloadMACIEFile( unsigned long handle,
                                      unsigned char slctMACIEs,
                                      const char*   regFile )
```

Summary

Download a sequence of register settings from the MACIE register file (.mrf) to the MACIE card. This can be used to initialize the MACIE card and power up the ASIC power supplies, in addition to other desired configuration options.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

regFile — MACIE register file .mrf – Input

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.16 MACIE_SetAcadiaAddressIncrement

```
MACIE_STATUS MACIE_GetAcadiaAddressIncrement ( unsigned long  handle,
                                               unsigned char  slctMACIEs,
                                               bool           bAutoAddrInc)
```

Summary

Set or clear the auto-increment option for mSPI read and write transactions to the ACADIA ASIC.

Parameters

handle Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

bAutoAddrInc Boolean value indicating the auto address increment will be set or not. If true, the auto address increment will be set; if false, the auto address increment will be not set – Output.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

When using the ACADIA ASIC, there is an option for choosing whether the ACADIA ASIC mSPI handler automatically increments or doesn't increment the address during mSPI register write and read transactions. The bAutoAddrInc value determines whether the auto-increment option is enabled or disabled.

By default, the auto-increment is enabled (controlled by register h0102 in the MACIE card). The default can be changed by including the MACIE register h0102 in the MACIE register load file, and setting its value to 0.

Disabling the auto-increment provides two benefits: It prevents the internal read-ahead of one address when performing an mSPI read (more correctly: the read-ahead still occurs but is done on the same address as the requested read), and it provides an option for reading or writing to the FIFOs using the mSPI block read and write functions.

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called. The function has no purpose when using the SIDECAR ASIC.

3.17 MACIE_GetAcadiaAddressIncrement

```
MACIE_STATUS MACIE_GetAcadiaAddressIncrement ( unsigned long  handle,
                                               unsigned char  slctMACIEs,
                                               bool*         bAutoAddrInc)
```

Summary

Returns the current auto-increment configuration for mSPI read and write transactions to the ACADIA ASIC.

Parameters

handle	Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.
slctMACIEs	8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.
bAutoAddrInc	Pointer to a boolean value indicating the auto address increment is set or not. If true, the auto address increment is set; if false, the auto address increment is not set – Output.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

The function returns the auto-increment configuration that can be set using the MACIE_SetAcadiaAddressIncrement() function (see function description) or using the MACIE register load file.

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called. Normally this function is called after loading MACIE register files. The function has no purpose when using the SIDECAR ASIC.

3.18 MACIE_WriteASICReg()

```
MACIE_STATUS MACIE_WriteASICReg( unsigned long  handle,
                                 unsigned char  slctASICs,
                                 unsigned short address,
                                 unsigned int   value,
                                 bool           bOption  )
```

Summary

Write value to ASIC register.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctASICs — 8-bit unsigned integer indicating the selected ASIC cards. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

address — 16-bit unsigned integer indicating a register address

value — 16-bit or 24-bit value which will be written to the register.

bOption — Boolean value indicating if DMA bit needs to be set (SIDECAR ASIC), or the mSPI-specific registers are to be addressed (ACADIA ASIC).

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.19 MACIE_ReadASICReg

```
MACIE_STATUS MACIE_ReadASICReg( unsigned long  handle,
                                unsigned char  slctASICs,
                                unsigned short address,
                                unsigned int   *value,
                                bool           b24bit,
                                bool           bOption  )
```

Summary

Read ASIC register value.

Parameters

handle Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctASICs 8-bit unsigned integer indicating the selected ASIC cards. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

address 16-bit unsigned integer indicating a register address. - Input

value Pointer to an 16-bit or 24-bit integer read from the register - Output

b24bit Boolean value indicating readback of 24 bit instead of 16-bit. – Input
This parameter is only applicable when using the SIDECAR ASIC.
For the ACADIA ASIC, this parameter is ignored.

bOption Boolean value indicating if DMA bit needs to be set (SIDECAR ASIC), or the mSPI-specific registers are to be addressed (ACADIA ASIC).

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.20 MACIE_WriteASICBlock

```
MACIE_STATUS MACIE_WriteASICBlock( unsigned long  handle,
                                   unsigned char  slctASICs,
                                   unsigned short address,
                                   unsigned int   *valueArray,
                                   int            arrSize,
                                   bool           bOption )
```

Summary

Write a number of contiguous ASIC registers starting at the specified register address.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctASICs — 8-bit unsigned integer indicating the selected ASIC cards. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

address — 16-bit unsigned integer indicating a register address. - Input

valueArray — Pointer to an unsigned integer array. The values in the array will be written to the ASIC registers starting at address - Input

arrSize — Array size indicating how many values will be written. -Input

bOption — Boolean value indicating if DMA bit needs to be set (SIDECAR ASIC), or the mSPI-specific registers are to be addressed (ACADIA ASIC).

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.21 MACIE_ReadASICBlock

```
MACIE_STATUS MACIE_ReadASICBlock( unsigned long  handle,
                                  unsigned char  slctASICs,
                                  unsigned short address,
                                  unsigned int   *valueArray,
                                  int            arrSize,
                                  bool           b24bit,
                                  bool           bOption )
```

Summary

Write a number of contiguous registers staring at the specified register address.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctASICs — 8-bit unsigned integer indicating the selected ASIC cards. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

address — 16-bit unsigned integer indicating a register address. - Input

valueArray — Pointer or address of an unsigned integer array storing the readback register values. - Output

arrSize — Array size indicating how many registers will be read. –Input

b24bit — Boolean value indicating readback of 24 bit instead of 16-bit. – Input
This parameter is only applicable when using the SIDECAR ASIC.
For the ACADIA ASIC, this parameter is ignored.

bOption — Boolean value indicating if DMA bit needs to be set (SIDECAR ASIC), or the mSPI-specific registers are to be addressed (ACADIA ASIC).

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.22 MACIE_DownloadASICFile

```
MACIE_STATUS MACIE_DownloadASICFile( unsigned long  handle,
                                     unsigned char  slctASICs,
                                     const char     *regFile,
                                     bool           bOption )
```

Summary

Download a sequence of values to the ASIC from an ASIC configuration file (like .mcd for SIDECAR or .ald for ACADIA).

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctASICs — 8-bit unsigned integer indicating the selected SIDECARs. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

regFile — SIDECAR firmware file .mcd – Input

bOption — Boolean value indicating if DMA bit needs to be set (SIDECAR ASIC), or the mSPI-specific registers are to be addressed (ACADIA ASIC).

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.23 MACIE_ClosePort

```
MACIE_STATUS MACIE_ClosePort( unsigned long  handle )
```

Summary

Close the port corresponding to the provided handle.

Parameters

None

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

When writing to or reading from the MACIE card through any of the interfaces (GigE / USB / CameraLink UART), the corresponding port is automatically opened to facilitate the communication. Afterwards, the port is kept open until the user calls the MACIE_ClosePort function to close the port. Typically, closing the port is only required if the port needs to made available to other applications on the computer.

3.24 MACIE_ResetErrorCounters

```
MACIE_STATUS MACIE_ResetErrorCounters( unsigned long handle,
                                       unsigned char slctMACIEs )
```

Summary

Reset all the MACIE error counters, including the MACIE and ASIC command error counters, timeout error counters and science interface error counters, etc.

Parameters

handle

Unsigned 32-bit integer obtained by calling MACIE_GetHandle function. - Input.

slctASICs

8-bit unsigned integer indicating the selected SIDECARs. This parameter can be obtained by calling MACIE_GetAvailableASICs. – Input.

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

If the MACIE card has detected any kind of error, it usually indicates the error condition with a flashing red LED. Calling this function should return all LEDs back to their normal state (typically green). The type and amount of detected errors can be obtained using the MACIE_GetErrorCounters() function.

3.25 MACIE_SetMACIEPhaseShift

```
MACIE_STATUS MACIE_SetMACIEPhaseShift( unsigned long  handle,
                                       unsigned char  slctMACIEs,
                                       unsigned short clkPhase )
```

Summary

Optimize the ASIC clock phase for science data transmission from ASIC to MACIE. Normally is function is only used for ASIC fast mode application.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctASICs — 8-bit unsigned integer indicating the selected SIDECARs. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

clkPhase — Bit 7-0: set phase shift for the ASIC clock if bit 8 is set.
Bit 8: enable ASIC phase shift, otherwise phase shift bits 7-0 are ignored

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.26 MACIE_GetMACIEPhaseShift

```
MACIE_STATUS MACIE_GetMACIEPhaseShift( unsigned long  handle,
                                       unsigned char  slctMACIEs,
                                       unsigned short *clkPhase )
```

Summary

Get the current ASIC clock phase shift setting from the MACIE card.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctASICs — 8-bit unsigned integer indicating the selected SIDECARs. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

clkPhase — Pointer to an unsigned integer indicating the ASIC clock phase setting. (See the bit description in MACIE_SetMACIEPhaseShift() function.

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.27 MACIE_DownloadLoadfile

```
MACIE_STATUS MACIE_DownloadLoadfile( unsigned long  handle,
                                     unsigned char  slctMACIEs,
                                     unsigned char  slctASICs,
                                     const char     *regFile,
                                     bool           bOption )
```

Summary

Download an individual register file or a master configuration file which includes a sequence of files to be loaded to the selected MACIEs and ASICs.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctASICs — 8-bit unsigned integer indicating the selected SIDECARs. If 0, the ASIC_Select parameter specified inside the file will be used. –Input

slctMACIEs — 8-bit unsigned integer indicating the selected MACIEs. If 0, the MACIE_Select parameter specified inside the file will be used. – Input

regFile — Individual register file or a script file name.

bOption — Boolean value indicating if DMA bit needs to be set (SIDECAR ASIC), or the mSPI-specific registers are to be addressed (ACADIA ASIC). This parameter will be overridden as soon as the first ASIC_MODE keyword is encountered in the load file (with optional modifiers DMA or MSPI_REG).

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

Example for a script file (test0.mcf)

```
MACIE_Select 1
LOAD    ./ load files/MACIE_Registers_tmp.mrf
WAIT 100
ASIC_Select 1
LOAD  ./load files/HxRG_Main.mcd
WAIT 200
```

Example for an individual load file (test1.glf)

```
MACIE_MODE
01b0 0081
```

```
01c2 03ff
01c3 07ff
ASIC_MODE DMA
4010 1234
4020 5555
ASIC_MODE
5000 4321
```

3.28 MACIE_GetErrorCounters

```
MACIE_STATUS MACIE_GetErrorCounters (unsigned long  handle,
                                     unsigned char  slctMACIEs,
                                     unsigned short *counterArray )
```

Summary

Read MACIE error counter registers.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected SIDECARs. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

counterArray — Pointer or address of an array allocated by the application. The array size should always be to MACIE_ERROR_COUNTERS. –Output.

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

If an error is detected, the MACIE card typically responds by flashing the red LED of the interface or ASIC where the error occurred. The list or errors returned is as follows:

List element	Error Group	SIDECAR ASIC	ACADIA ASIC
1	UART Port	Parity Errors	Parity Errors
2		Stopbit Errors	Stopbit Errors
3		Timeout Errors	Timeout Errors
4	USB Port	Timeout Errors	Timeout Errors
5	GigE Port	Timeout Errors	Timeout Errors
6	ASIC 1 Configuration	Acknowledge Errors Type 1	mSPI CRC Errors
7		Acknowledge Errors Type 2	mSPI Start Errors
8		Start Bit Timeout Errors	mSPI TxFIFO Errors
9		Stop Bit Timeout Errors	mSPI RxFIFO Errors
10		Stop Bit Errors	n/a
11		Parity Errors	n/a
12		Data Errors	n/a
13		CRC errors	n/a

List element	Error Group	SIDECAR ASIC	ACADIA ASIC
14	ASIC 2 Configuration	Acknowledge Errors Type 1	mSPI CRC Errors
15		Acknowledge Errors Type 2	mSPI Start Errors
16		Start Bit Timeout Errors	mSPI TxFIFO Errors
17		Stop Bit Timeout Errors	mSPI RxFIFO Errors
18		Stop Bit Errors	n/a
19		Parity Errors	n/a
20		Data Errors	n/a
21		CRC Errors	n/a
22	Main Science FIFO	Science FIFO Overflow Errors	Science FIFO Overflow Errors
23		spare	spare
24		spare	spare
25		spare	spare
26	ASIC 1 Science Data	Stop Errors	Stop Errors
27		Parity Errors	Parity Errors
28		Data Errors	Data Errors
29		CRC Errors	CRC Errors
30	ASIC 2 Science Data	Stop Errors	Stop Errors
31		Parity Errors	Parity Errors
32		Data Errors	Data Errors
33		CRC Errors	CRC Errors

3.29 MACIE_ConfigureCamLinkInterface

```
MACIE_STATUS MACIE_ConfigureCamLinkInterface( unsigned long  handle,
                                              unsigned char  slctMACIEs,
                                              unsigned short mode,
                                              const char     *dcfFFile,
                                              unsigned short timeout,
                                              unsigned short frameX,
                                              unsigned short frameY,
                                              short          *bufferSize )
```

Summary

Set up Camera Link interface for image acquisition.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected SIDECARs. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

mode — bit<4> Send Dummy Frames

bit<6-5> Select dummy frame type:
b00: all values are 0
b01: incrementing value, starts at 0 at the beginning of each row
b10: incrementing value, continues to increment at beginning of each row (instead of resetting to 0)
b11: constant value per row, value increments with each row

Note: This input parameter has been added in release V5.1. Older programs or scripts using this function may have to be updated accordingly.

dcfFFile — Camera Link configuration file (.dcf) – Input

timeout — 16 bit unsigned integer (in 100 us steps) after which the remainder of the Camera Link frame is filled with dummy 0s by MACIE card. If 0, the default timeout of 1000 (in 100 us steps) will be used.-Input

frameX — Camera Link image size X -Input

frameY — Camera Link image size Y -Input

bufferSize — Pointer to an integer indicating the maximum number of images which can be stored in the non-paged memory allocated in the MIL Config. - Output

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.

If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

Note: This function is tied to using the Matrox Solios or Helios frame grabber with the MIL-lite library. If using other frame grabbers, the necessary MACIE setup has to be performed by the user using direct MACIE register writes to addresses 0x01c0 - 0x01c4.

3.30 MACIE_ConfigureGigeScienceInterface

```
MACIE_STATUS MACIE_ConfigureGigeScienceInterface( unsigned long  handle,
                                                  unsigned char  slctMACIEs,
                                                  unsigned short mode,
                                                  int            frameSize,
                                                  unsigned short remotePort,
                                                  int            *bufSize )
```

Summary

Set up GigE science data interface for image acquisition.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected SIDECARs. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

mode — bit <1-0> GigE_DataFormat:

0 = 16-bit words
1 = 24-bit words (e.g. 2*12bit)
2 = 32-bit words
3 = 32-bit words (2*12 bit aligned on word boundary)

bit <6-4> Dummy Frame Type (auto-generates test science data):

b000: Dummy test frames disabled (normal mode of operation)
b001: fixed value of 0 for the whole frame
b010: incrementing value, starts at 0 at the start of each row
b011: incrementing value, continues to increment at start of each row (instead of resetting to 0)
b100: constant value per row, value increments with each row;
b101: fixed value for the whole frame, increments by 256 with each frame (512 for GigE_DataFormat 1 and 3)
b110: incrementing value, starts over at the beginning of each row, increments by 256 with each frame (512 for GigE_DataFormat 1 and 3)
b111: incrementing value, continues to increment at beginning of each row, increments by 256 with each frame (512 for GigE_DataFormat 1 and 3)

frameSize — Integer indicating the image size of (frameX * frameY). When intending to read data using the MACIE_ReadGigeScienceData() function instead of the MACIE_ReadGigEScienceFrame() function, set this parameter to 0.

remotePort — 16 bit unsigned integer indicating the GigE port number (e.g. 42037) -Input

bufferSize — Pointer to an integer indicating the available buffer size in the low level operating system buffer for the TCP socket connection [in KB]. This is for

user information purposes only. Refer to the User Manual for MACIE Acquisition Control for information on how to increase this buffer on Linux.
- Output

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

If only full frames are to be acquired, set the frameSize parameter to match the size of the frame (in number of pixels), and use the MACIE_ReadGigEScienceFrame() to retrieve the capture frames. If other types of data are to be read (e.g. varying frame sizes or blocks of data), set the frameSize parameter to 0 and use the MACIE_ReadGigeScienceData() function to retrieve the captured data.

3.31 MACIE_ConfigureUSBScienceInterface

```
MACIE_STATUS MACIE_ConfigureUSBScienceInterface(  unsigned long   handle,
                                                  unsigned char   slctMACIEs,
                                                  unsigned short  mode,
                                                  int             frameSize,
                                                  short           uBuffers )
```

Summary

Set up USB science data interface for image acquisition.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected SIDECARs. This parameter can be obtained by calling MACIE_GetAvailableASICs(). – Input.

mode — bit <1-0> USB_DataFormat:

0 = 16-bit words
1 = 24-bit words (e.g. 2*12bit)
2 = 32-bit words
3 = 32-bit words (2*12 bit aligned on word boundary)

bit <6-4> Dummy Frame Type (auto-generates test science data):

b000: Dummy test frames disabled (normal mode of operation)
b001: fixed value of 0 for the whole frame
b010: incrementing value, starts at 0 at the start of each row
b011: incrementing value, continues to increment at start of each row (instead of resetting to 0)
b100: constant value per row, value increments with each row;
b101: fixed value for the whole frame, increments by 256 with each frame (512 for USB_DataFormat 1 and 3)
b110: incrementing value, starts over at the beginning of each row, increments by 256 with each frame (512 for USB_DataFormat 1 and 3)
b111: incrementing value, continues to increment at beginning of each row, increments by 256 with each frame (512 for USB_DataFormat 1 and 3)

bit <8> Dual Pipe Mode: Configures separate USB Pipe for science data instead of sharing the same Pipe with configuration read data. Enables parallel register read and science data acquisition, at the cost of reduced bandwidth. Requires to configure the USB control chip accordingly, which will be taken care of when using the corresponding MACIE library function.

- Input.

frameSize — Integer indicating the image size or internal buffer size in number of data words. When intending to read data using MACIE_ReadUSBScienceData() instead of MACIE_ReadUSBScienceFrame(), this parameter only indicates the buffer size. - Input

nBuffers — Number of image buffers allocated to receive science data. - Input
Note: The buffer size is equal to the frameSize.

Return Value

MACIE_OK if successful, MACIE_FAIL if not successful.
If MACIE_FAIL is returned, call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

If only full frames are to be acquired, set the frameSize parameter to match the size of the frame (in number of pixels), and use the MACIE_ReadUSBScienceFrame() to retrieve the captured frames. If other types of data are to be read (e.g. varying frame sizes or blocks of data), set the frameSize parameter to a reasonable buffer size (for example: 4194394), and then use the MACIE_ReadUSBScienceData() function to retrieve the data.

Important:
This function can configure the science data interface in either single-pipe mode or dual pipe mode (see bit 8 of mode parameter). In single pipe mode, the science data interface and the configuration data interface share the same USB pipe. Once the science data interface is configured, all the read register functions (for example: MACIE_ReadASICReg) using the USB interface should not be called anymore until the science data interface is closed (MACIE_CloseCamlinkScienceInterface). Otherwise, data corruption may occur. In dual pipe mode, science data interface and configuration data interface are using separated pipes, and which supports simultaneous reading of science data and configuration registers.
Note: Since dual-pipe mode requires sharing bandwidth between a science data pipe and a configuration data pipe over USB, the maximum science data bandwith is reduced. Therefore, dual-pipe mode should only be used if the required bandwidth is well below the maximum possible value of 340MByte/s (approx. maximum in dual-pipe mode is 200MByte/s, exact limit has to be determined experimentally).

3.32 MACIE_AvailableScienceData

```
unsigned long MACIE_AvailableScienceData( unsigned long handle )
```

Summary

Return the number of science data bytes available on the specified interface port.

Parameters

handle	Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

Return Value

Return the number of bytes available on the port.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called. In addition, the interface has to be first configured for science data acquisition using the MACIE_ConfigureGigeScienceInterface() or MACIE_ConfigureUSBScienceInterface() functions.

This function can be used for the GigE and USB interface. It allows the user to check when data is available on the port after triggering the image ramp acquisition (i.e. the image ramp has started).
For the Camera Link interface, this function is not useful since the image is transferred frame-by frame instead of byte-by-byte.

3.33 MACIE_AvailableScienceFrames

```
unsigned long MACIE_AvailableScienceFrames( unsigned long handle )
```

Summary

Return the number of frames available on the specified interface port.

Parameters

handle	Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

Return Value

Return the number of frames available on the port.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called. In addition, the desired interface has to be first configured for science data acquisition using the MACIE_ConfigureGigeScienceInterface(), MACIE_ConfigureUSBScienceInterface(), or MACIE_ConfigureCamLinkInterface() functions.

This function is only applied for GigE and Camera Link Interfaces. For USB interface, please use MACIE_AvailableScienceData to check the number of available science data.

3.34 MACIE_ ReadGigeScienceFrame

```
unsigned short* MACIE_ReadGigeScienceFrame( unsigned long handle,
                                            unsigned short timeout )
```

Summary

Read image from the specified interface port.

Parameters

handle	Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. -Input.
timeout	16 bit unsigned integer (in ms) after which the function will stop reading from the port and return NULL. -Input

Return Value

If successful: returns a pointer to a frame-sized array of unsigned short image data.

If not successful: returns NULL. Call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

In addition, the interface has to be first configured for science data acquisition using the MACIE_ConfigureGigeScienceInterface() function In addition, and the frameSize parameter of the MACIE_ConfigureGigeScienceInterface() function has to be set to the correct frame size (in number of pixels).

3.35 MACIE_ ReadUSBScienceFrame

```
unsigned short* MACIE_ReadUSBScienceFrame( unsigned long handle,
                                           unsigned short timeout)
```

Summary

Read image from the specified interface port.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. -Input.

timeout — 16 bit unsigned integer (in ms) after which the function will stop reading from the port and return NULL. –Input

Return Value

If successful: returns a pointer to a frame-sized array of unsigned short image data.

If not successful: returns NULL. Call the MACIE_Error() function for details regarding the type of failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

In addition, the interface has to be first configured for science data acquisition using the MACIE_ConfigureUSBScienceInterface() function, with the frameSize parameter set to the correct frame size (in number of pixels).

3.36 MACIE_ ReadCamlinkScienceFrame

```
unsigned short* MACIE_ReadCamlinkScienceFrame( unsigned long  handle,
                                               char           *tifFileName,
                                               unsigned short timeout )
```

Summary

Read Camera Link image.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. -Input.

tifFileName — .tif file name which is used by the MIL library to save the image data to disk. If an empty string is provided, no .tif file will be saved to disk.

timeout — 16 bit unsigned integer (in ms) after which the function will stop reading from the port and return NULL. -Input

Return Value

If successful: return an pointer to an frame-sized array of integer – image data.

If failed: return NULL. Call MACIE_Error for the detail of the failure.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called. In addition, the interface has to be first configured for science data acquisition using the MACIE_ConfigureCamLinkInterface() function.

3.37 MACIE_WriteFitsFile

```
MACIE_STATUS MACIE_WriteFitsFile( char           *fileName,
                                  unsigned short frameX,
                                  unsigned short frameY,
                                  unsigned short *pData,
                                  unsigned short nHeaders,
                                  MACIE_FitsHdr  *pHeaders )
```

Summary

Write image data to .fits file.

Parameters

fileName	fits file name -Input
frameX	Image size X - Input
frameY	Image size Y – Input
pData	Pointer to the image data array - Input
nHeaders	number of fits header units (number of elements in the pHeader array) - Input
pHeaders	pointer to an array of MACIE_FitsHdr structures. - Input

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

If a custom header is to be added to the fits file, the MACIE_FitsHdr structure array has to be populated first. Please refer to the macie.h file for the structure definition. Three data types can be added: int, float, and string. If int is specified as the value type, both the float and string parameter do not have to be assigned. Likewise if float or string is specified, the other two parameters do not have to be assigned.

Also, for detailed information on the general fits header formatting, please refer to the website https://fits.gsfc.nasa.gov/fits_primer.html.

3.38 MACIE_ ReadGigeScienceData

```
int MACIE_ReadGigeScienceData( unsigned long  handle,
                               unsigned short timeout,
                               long           n,
                               unsigned short *pData )
```

Summary

Read science data. This function can be used for capturing science data in a non-fixed frame size format (for example: when interleaving guide windows with full field data).

Parameters

handle Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. -Input.

timeout 16 bit unsigned integer (in ms) after which the function will stop reading from the port and return NULL. –Input

n Number of science data words to read – Input

pData Pointer to a pre-allocated unsigned short data buffer. The buffer has to have sufficient memory allocated to hold at least n words. Function will fill received science data into this buffer. – Output.

Return Value

Number of acquired science data words.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

In addition, the interface has to be first configured for science data acquisition using the MACIE_ConfigureGigeScienceInterface() function, and the frameSize parameter of the MACIE_ConfigureGigeScienceInterface() function has to be set to 0.

3.39 MACIE_ ReadUSBScienceData

```
int MACIE_ReadUSBScienceData( unsigned long  handle,
                              unsigned short timeout,
                              long           n,
                              unsigned short *pData )
```

Summary

Read science data. This function can be used for capturing science data in a non-fixed frame size format (for example: when interleaving guide windows with full field data).

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

timeout — 16 bit unsigned integer (in ms) after which the function will stop reading from the port and return. – Input

n — Number of science data words to read – Input

pData — Pointer to a pre-allocated unsigned short data buffer. The buffer has to have sufficient memory allocated to hold at least n words. Function will fill received science data into this buffer. – Output.

Return Value

Number of acquired science data words.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

In addition, the interface has to be first configured for science data acquisition using the MACIE_ConfigureGigeScienceInterface() function, and the frameSize parameter of the MACIE_ConfigureGigeScienceInterface() function has to be set to 0.

3.40 MACIE_CloseCamlinkScienceInterface

```
MACIE_STATUS MACIE_CloseCamlinkScienceInterface( unsigned long handle,
                                                 unsigned char slctMACIEs);
```

Summary

Close Solios frame grabber card and disable MACIE Camera Link Science interface.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called. Normally this function should be called after the image acquisition is done.

3.41 MACIE_CloseGigeScienceInterface

```
MACIE_STATUS MACIE_CloseGigeScienceInterface( unsigned long handle,
                                              unsigned char slctMACIEs);
```

Summary

Close MACIE GigE Science interface.

Parameters

handle Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called. Normally this function should be called after the image acquisition is done.

3.42 MACIE_CloseUSBScienceInterface

```
MACIE_STATUS MACIE_CloseUSBScienceInterface( unsigned long handle,
                                             unsigned char slctMACIEs);
```

Summary

Close MACIE USB Science interface.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called. Normally this function should be called after the image acquisition is done.

3.43 MACIE_SetVoltage

```
MACIE_STATUS MACIE_SetVoltage( unsigned long handle,
                               unsigned char slctMACIEs,
                               MACIE_PWR_DAC powerName,
                               float         powerValue)
```

Summary

Set voltage (v) or current (mA) to the power supply item listed in the structure of MACIE_PWR_DAC.

Parameters

handle Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

powerName A power item of enum structure of MACIE_PWR_DAC – Input.

powerValue floating number indicating voltage in the unit of V and the current in unit of mA – Input.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.44 MACIE_GetVoltage

```
MACIE_STATUS MACIE_GetVoltage( unsigned long handle,
                               unsigned char slctMACIEs,
                               MACIE_PWR_DAC powerName,
                               Float*        powerValue)
```

Summary

Get voltage (V) or current (mA) setting for the given power supply item listed in the structure of MACIE_PWR_DAC.

Parameters

handle	Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.
slctMACIEs	8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.
powerName	Power item of enum structure of MACIE_PWR_DAC – Input.
powerValue	Pointer to a floating number indicating voltage in the unit of V and the current in unit of mA - Output

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.45 MACIE_EnablePower

```
MACIE_STATUS MACIE_EnablePower( unsigned long   handle,
                                unsigned char   slctMACIEs,
                                MACIE_PWR_CTRL* pwrCtrlArr,
                                short           n)
```

Summary

Enable MACIE supply voltages using the power control items listed in the structure of MACIE_PWR_CTRL.

Parameters

handle Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

pwrNameArr Pointer to an array of power control items listed in the enum structure of MACIE_PWR_CTRL – Input.

N Number of power control items in the pwrNameArr. – Input.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.46 MACIE_ DisablePower

```
MACIE_STATUS MACIE_DisablePower( unsigned long    handle,
                                unsigned char    slctMACIEs,
                                MACIE_PWR_CTRL*  pwrCtrlArr,
                                short            n)
```

Summary

Disable the power controls for the power control items listed in the structure of MACIE_PWR_CTRL.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

pwrNameArr — Pointer to an array of power control items listed in the enum structure of MACIE_PWR_CTRL – Input.

n — Number of power control items in the pwrNameArr. – Input.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.47 MACIE_ SetPower

```
MACIE_STATUS MACIE_SetPower( unsigned long   handle,
                             unsigned char   slctMACIEs,
                             MACIE_PWR_CTRL  pwrCtrlName,
                             bool            bEnablePower)
```

Summary

Enable or disable a single power control item listed in the structure of MACIE_PWR_CTRL.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

pwrCtrlName — Enum item listed in MACIE_PWR_CTRL, indicating the power control item – Input.

bEnablePower — Boolean to indicate enable or disable the power control. If true, enable the power control, otherwise disable the power control.- Input

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.48 MACIE_GetPower

```
MACIE_STATUS MACIE_GetPower( unsigned long   handle,
                             unsigned char   slctMACIEs,
                             MACIE_PWR_CTRL  pwrCtrlName,
                             bool*           bEnablePower)
```

Summary

Get the power control status for the given power control item listed in the structure of MACIE_PWR_CTRL. The status is indicated by the output parameter bEnablePower.

Parameters

handle	Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs	8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

pwrCtrlName	enum item listed in MACIE_PWR_CTRL, indicating the power control item – Input.

bEnablePower	Pointer to a boolean value to indicate if the specified power control is enabled or not. If true, enable the power control, otherwise disable the power control. – Output

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.49 MACIE_SetTelemetryConfiguration

```
MACIE_STATUS MACIE SetTelemetryConfiguration( unsigned long              handle,
                                              unsigned char              slctMACIEs,
                                              MACIE_TLM_SAMPLE_RATE      vSampleRate,
                                              MACIE_TLM_AVERAGE          vAverage,
                                              MACIE_TLM_SAMPLE_RATE      iSampleRate,
                                              MACIE_TLM_AVERAGE          iAverage,
                                              MACIE_TLM_GROUND_REFERENCE groundRef)
```

Summary

Set sample rate, average parameters and ground references for telemetry measurement by MACIE card.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

vSampleRate — Sample rate for voltage measurement, selectable from any value in enum MACIE_TLM_SAMPLE_RATE. – Input.

vAverage — Average parameter for voltage measurement, selectable from any value in enum MACIE_TLM_AVERAGE. – Input.

iSampleRate — Sample rate fo current measurement, selectable from any value in enum MACIE_TLM_SAMPLE_RATE. – Input.

iAverage — Average parameter for current measurement, selectable from any value in enum MACIE_TLM_AVERAGE. – Input.

groundRef — Ground reference selected for the voltage / current measurement, selectable from any value in enum MACIE_TLM_GROUND_REFERENCE. – Input.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.50 MACIE_GetTelemetryConfiguration

```
MACIE_STATUS MACIE GetTelemetryConfiguration( unsigned long      handle,
                                     unsigned char               slctMACIEs,
                                     MACIE_TLM_SAMPLE_RATE*      vSampleRate,
                                     MACIE_TLM_AVERAGE*          vAverage,
                                     MACIE_TLM_SAMPLE_RATE*      iSampleRate,
                                     MACIE_TLM_AVERAGE*          iAverage,
                                     MACIE_TLM_GROUND_REFERENCE* groundRef)
```

Summary

Get sample rate, average parameters and ground references used for telemetry measurement by MACIE card.

Parameters

handle
Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs
8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

vSampleRate
Pointer for the sample rate for voltage measurement, containing an item from the enum MACIE_TLM_SAMPLE_RATE. – Output.

vAverage
Pointer for the average parameter for voltage measurement, containing an item from the enum MACIE_TLM_AVERAGE. – Output.

iSampleRate
Pointer for the sample rate for current measurement, containing an item from the enum MACIE_TLM_SAMPLE_RATE. – Output.

iAverage
Pointer for the average parameter for current measurement, containing an item from the enum MACIE_TLM_AVERAGE. – Output.

groundRef
Pointer for the ground reference for the voltage/current measurement, containing an item from the enum MACIE_TLM_GROUND_REFERENCE. – Output.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.51 MACIE_GetTelemetry

```
MACIE_STATUS MACIE GetTelemetry( unsigned long   handle,
                                 unsigned char   slctMACIEs,
                                 MACIE_TLM_ITEM  tlmId,
                                 float*          tlmValue)
```

Summary

Get telemetry measurement performed by MACIE card for the given MACIE_TLM_ITEM item.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

tlmId — An enum value which is listed in MACIE_TLM_ITEM. – Input.

tlmValue — Pointer for a floating number indicating the voltage measurement (V) or current measurement (mA). – Output.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

3.52 MACIE_GetTelemetrySet

```
MACIE_STATUS MACIE_GetTelemetrySet( unsigned long   handle,
                                    unsigned char   slctMACIEs,
                                    MACIE_TLM_ITEM* tlmIdArr,
                                    short           n,
                                    float*          tlmValArr)
```

Summary

Get a set of telemetry measurements performed by the MACIE card for the given array of telemetry items listed in the enum structure of MACIE_TLM_ITEM.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

tlmIdArr — The address of an array of telemetry items listed in the enum structure of MACIE_TLM_ITEM. – Input.

n — Number of telemetry items in the tlmIdArr. – Input.

tlmValArr — Pointer to a floating array for storing the measured telemetry results. The array has to be pre-allocated with at least n elements. – Output.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

Depending on the number of telemetry items (n) to be measured, and the telemetry configuration set by the user, this function might take up to 30s to complete. As an example, with the default telemetry configuration of averaging and rate, it will take about ~5 seconds to measure the whole telemetry set.

3.53 MACIE_GetTelemetryAll

```
MACIE_STATUS MACIE_GetTelemetryAll( unsigned long  handle,
                                    unsigned char  slctMACIEs,
                                    float*         pTlmVals)
```

Summary

Get the full set of all telemetry measurements (total of 79 items) performed by the MACIE card for the given array of telemetry items listed in the enum structure of MACIE_TLM_ITEM.

Parameters

handle — Unsigned 32-bit integer obtained by calling MACIE_GetHandle() function. - Input.

slctMACIEs — 8-bit unsigned integer indicating the selected MACIE cards. This parameter can be obtained by calling MACIE_GetAvailableMACIEs(). – Input.

pTlmVals — Pointer to a floating array storing the telemetry measurement results. The array has to be pre-allocated with a size of at least 79 elements. – Output.

Return Value

MACIE_OK if successful. MACIE_FAIL if failed.

Comments

This function can only be called after MACIE_CheckInterfaces() and MACIE_GetHandle() have been called.

Depending on the telemetry configuration set by the user, this function might take up to 30s to complete. As an example, with the default telemetry configuration of averaging and rate, it will take about ~5 seconds to measure the whole telemetry set.