OOIHIP

Handheld Interface Package Programmer's Guide

Version 1.0

Ocean Optics, Inc.
380 Main Street
Dunedin, FL 34698
(727) 733-2447
(727) 733-3962 fax

For the latest information, consult our web site:
www.OceanOptics.com

Or, e-mail our Technical Service Department:
TechSupport@OceanOptics.com

012800

Trademarks

Microsoft, C++, Visual Basic, Windows NT, Handheld PC Professional are registered trademarks of the Microsoft Corporation.

Limit of Liability

Every effort has been made to make this manual as complete and as accurate as possible. The information and code provided hereunder (collectively referred to as “software”) is provided on an “as is” basis. No warranty or fitness of any kind in regard to this manual or the software code is implied. In no event shall Ocean Optics, Inc. or its suppliers be liable or responsible due to any loss or damages arising from the information contained in this manual or from the OOIHIP product.

Contents

Description of Spectral Acquisition Functions (continued)

Description of Graphics Functions (continued)

Description of Communications Functions (continued)

Overview

The Ocean Optics, Inc. Handheld Interface Package (OOIHIP) is a software driver package for Windows CE that allows you to write custom software for Ocean Optics spectrometers. Complete source code is provided to ease your software development tasks.

The OOIHIP includes libraries for serial communication, for data acquisition from Ocean Optics spectrometers, for high-performance spectral plotting, for linear and polynomial regressions, and for several utility functions that are not included in the Windows CE run-time library. All source code is provided in flat C-code. DLLs generated from the source code are provided for all popular Windows CE processors.

The OOIHIP is an extremely powerful library. OOIPS2000, the spectrometer operating software designed for our Palm-SPEC Spectrophotometer, was written using functions from this Dynamic Link Library.

Software License

This Software gives you the ability to write applications to acquire and process data from Ocean Optics spectrometers. You have the right to redistribute the libraries contained in the Software, subject to the following conditions:

1. You are developing applications to control Ocean Optics spectrometers. If you use the code contained herein to develop applications for other purposes, you MUST obtain a separate software license.
2. You distribute only the drivers necessary to support your application.
3. You place all copyright notices and other protective disclaimers and notices contained on the Software on all copies of the Software and your software product.
4. You notify Ocean Optics annually of how many copies of the driver you distribute.
5. You or your company provides technical support to the users of your application. Ocean Optics, Inc. will not provide software support to these customers.
6. You agree to indemnify, hold harmless, and defend Ocean Optics, Inc. and its suppliers from and against any claims or lawsuits, including attorneys' fees that arise or result from the use or distribution of your software product and any modifications to the Software.
7. You distribute the software only in compiled form. The source code distributed with the OOIHIP is intended for use only as a software development tool.

System Requirements

In order to use the libraries contained in the OOIHIP, you must have installed on your system:

- Microsoft Visual C++ 6.0 or greater
- Microsoft Visual C++ Toolkit for Windows CE
- Windows CE Software Development Kit (SDK) for the Palm-size PC (Version 2.11) or the Handheld PC Pro (Version 3)
- Microsoft Visual C++ Service Pack 2 (**Service Pack 3 is incompatible with the Windows CE Toolkit)**
- A handheld or palm-size PC running Windows CE (Palm-size version 2.11 or HPC Pro version 3)

Writing applications using Visual Basic is not directly supported by the OOIHIP. Visual Basic for Windows CE has a severe limitation in that applications cannot pass structures to functions in a DLL, or receive floating point return values from these functions.

Installation

To install the software, place the OOIHIP Installation Disk into a floppy drive and run the SETUP.EXE program. An installation wizard guides you through the rest of the setup program and gives you the option to install some or all of the sample applications.

The Basics

The OOIHIP allows you to easily control all aspects of data acquisition, digital I/O and triggering functions for the Ocean Optics line of miniature fiber optic spectrometers. There are, however, several requirements that must be met before you begin to write your own program:

1. You must have either an Ocean Optics SAD500 Serial Port Interface, a USB2000 (operating in RS-232 mode) or a Palm-SPEC Spectrophotometer. If you are using some other device as an A/D converter, you cannot use these drivers.
2. You must have an Ocean Optics spectrometer. The driver controls the acquisition of data from the following Ocean Optics spectrometers: S2000, S1024DW, S1024DWX, USB2000 and Palm-SPEC. However, these spectrometers must be supported through the proper A/D device.

The driver file (OOIHIP.DLL) must be located in a directory where your application can find it. Appropriate directories include the directory of your program, or the WINDOWS directory. We strongly recommended that you place only one copy of the driver in either location. All Ocean Optics applications install drivers in the installation directory for that specific application.

Constants

Communications Error Codes – OOIComCE Functions

Header: OOIComCE.h

Symbol	Value	Use
ERROR_NO_ERROR	0	No error was generated by the last command
ERROR_NOT_OPENED	-1	The communications port was not opened prior to calling the functions
ERROR_CANT_GET_DCB	-2	An error occurred while the library was attempting to retrieve the device control block from the system
ERROR_CANT_SET_DCB	-3	An error occurred while the library was attempting to set the device control block from the system
ERROR_CANT_GET_COUNT	-4	An error occurred while trying to retrieve the number of bytes in either the transmit or receive buffer
ERROR_WRITE_FAILED	-5	A call to OOIComCE_WriteBuffer was not successful

Communications Error Codes – Windows CE Errors

Header: OOIComCE.h

Symbol	Value	Use
ERROR_BADID	-1	The device identifier is invalid or unsupported
ERROR_BAUDRATE	-12	The device's baud rate is unsupported
ERROR_BYTESIZE	-11	The specified byte size is invalid
ERROR_DEFAULT	-5	The default parameters are in error
ERROR_HARDWARE	-10	The hardware is not available (locked by another device)
ERROR_MEMORY	-4	The function cannot allocate the queues
ERROR_PORT_NOT_OPENED	-3	The device is not open
ERROR_OPENED	-2	The device is already open
ERROR_BREAK	0x0010	The hardware detected a break condition
ERROR_DNS	0x0800	The parallel device is not selected
ERROR_FRAME	0x0008	The hardware detected a framing error
ERROR_IOE	0x0400	An input/output (I/O) error occurred
ERROR_MODE	0x8000	A requested mode is not supported, or the hPort parameter is invalid. If this flag is set, it is the only valid error
ERROR_OOP	0x1000	The parallel device signaled that it is out of paper
ERROR_OVERRUN	0x0002	Character-buffer overrun. The next character is lost
ERROR_PTO	0x0200	Time-out on a parallel device
ERROR_RXOVER	0x0001	Receive queue overflow. There is either no room in the receive queue, or a character was received after the EOF character was received
ERROR_RXPARITY	0x0004	The hardware detected a parity error
ERROR_TXFULL	0x0100	The _VCOMM_WriteComm service was called when the transmit queue was full

These constants are returned by the OOIComCE_GetCommError function and indicate the nature of the communication error.

Communications Events

Header: OOIComCE.h

Symbol	Value	Use
EVENT_BREAK	0x0040	A break was detected on input
EVENT_CTS	0x0008	The CTS (clear-to-send) signal changed state
EVENT_DSR	0x0010	The DSR (data-set-ready) signal changed state
EVENT_ERR	0x0080	A line-status error occurred
EVENT_PERR	0x0200	A printer error occured
EVENT_RING	0x0100	A ring indicator was detected
EVENT_RLSD	0x0020	The RLSD (receive-line-signal-detect) signal changed state
EVENT_RX80FULL	0x0400	The receive buffer is 80 percent full
EVENT_RXCHAR	0x0001	A character was received and placed in the input buffer
EVENT_RXFLAG	0x0002	The event character was received and placed in the input buffer
EVENT_TXEMPTY	0x0004	The last character in the output buffer was sent

These constants are returned by the OOIComCE_GetEventMask, OOIComCE_SetEventMask and the OOIComCE_WaitCommEvent functions to specify communication events and event notifications.

Communications Parity

Header: OOIComCE.h

Symbol	Value	Use
PARITY_NONE	0	Use no parity
PARITY_ODD	1	Use odd parity
PARITY_EVEN	2	Use even parity
PARITY_MARK	3	Use mark parity
PARITY_SPACE	4	Use space parity

These constants are used by the OOIComCE_SetCommunicationsParameters to specify the parity of the serial communications.

Communications Ports

Header: OOIComCE.h

Symbol	Value	Use
COM1	1	Use the computer’s first serial port
COM2	2	Use the computer’s second serial port
COM3	3	Use the computer’s third serial port
COM4	4	Use the computer’s fourth serial port

These constants are used by the OOIComCE_Open functions to specify the serial port opened.

Communications Stop Bits

Header: OOIComCE.h

Symbol	Value	Use
STOPBITS_1	0	Use one stop bit
STOPBITS_1_5	1	Use one and one-half stop bits
STOPBITS_2	2	Use two stop bits

These constants are used by the OOIComCE_SetCommunicationParameters function to specify the number of stop bits used in the serial communications.

Graph Axes

Header: OOIGraphCE.h

Symbol	Value	Use
X_AXIS	0	Autoscales the x-axis
Y_AXIS	1	Autoscales the y-axis

These constants are used by the OOIGraphCE_AutoscaleAxis function.

Graph Cursor Types

Header: OOIGraphCE.h

Symbol	Value	Use
CURSOR_NONE	0	No cursor is drawn
CURSOR_X	1	Only the vertical cursor is drawn
CURSOR_Y	2	Only the horizontal cursor is drawn
CURSOR_BOTH	3	Both the horizontal and vertical cursors are drawn

These constants are used by the OOIGraphCE_ShowCursor function.

Graph Grid Line Types

Header: OOIGraphCE.h

Symbol	Value	Use
SOLID_LINE	0	Grids are drawn as solid lines
DASHED_LINE	1	Grids are drawn as dashed lines

The OOIGraphCE_SetGridLineType uses these constants

Graph Line Type

Header: OOIGraphCE.h

Symbol	Value	Use
POINT	0	Draws the graph trace as points
LINES	1	Draws the graph trace as lines

These constants are used by the OOIGraphCE_SetTraceType function.

Graph Pixels

Header: OOIGraphCE.h

Symbol	Value	Use
PIXELS	0	Number of pixels displayed in the graph

This constant sets the number of x-y pairs that are displayed in the graph window.

Graph Point Validation

Header: OOIGraphCE.h

Symbol	Value	Use
XANDY	0	Point is in both x and y extents of the graph
XONLY	1	Point is only in the x extent of the graph
YONLY	2	Point is only in the y extent of the graph
NEITHER	3	Point is neither in the x or y extents of the graph

These constants are returned by the OOIGraphCE_IsPointInGraph function.

SAD500 Baud Rates

Header: OOISADCE.h

These constants are used by SAD_Init and SAD_SetBaudRate to indicate to the driver the baud rate at which the SAD500 should communicate.

Symbol	Value	Use
SAD_BAUD_2400	0	2400 baud
SAD_BAUD_4800	1	4800 baud
SAD_BAUD_9600	2	9600 baud (default)
SAD_BAUD_19200	3	19200 baud
SAD_BAUD_38400	4	38400 baud
SAD_BAUD_57600	5	57600 baud
SAD_BAUD_115200	6	115200 baud (not supported in OOIDRV16.DLL)

SAD500 Close Commands

Header: OOISADCE.h

These commands are passed to SAD_Close to specify which acquisition parameters are stored in permanent memory.

Symbol	Value	Use
SAD_CLOSE_NO_SAVE	0	Saves no parameters
SAD_CLOSE_STORE_ALL_BUT_BAUD	1	Stores all acquisition parameters except baud rate
SAD_CLOSE_STORE_ALL	2	Stores all acquisition parameters including baud rate
SAD_CLOSE_STORE_BAUD_DEFAULT	3	Stores all acquisition parameters and sets the baud rate to the default of 9600

SAD500 Correlated Double Sampling (CDS) Modes

Header: OOISADCE.h

When used with the S1024DW, the SAD500 can use correlated double sampling to automatically correct for electrical offset of each pixel.

Symbol	Value	Use
SAD_CDS_OFF	0	Do not use correlated double sampling
SAD_CDS_ON	1	Use correlated double sampling

SAD500 Data Compression Modes

Header: OOISADCE.h

Configure the SAD500 to transmit compressed spectral information to speed data transfer rates.

Symbol	Value	Use
SAD_COMPRESSION_OFF	0	Do not use data compression
SAD_COMPRESSION_ON	1	Use data compression

SAD500 Data Checksum Modes

Header: OOISADCE.h

To ensure data integrity, configure the SAD500 to transmit a checksum of all spectral data.

Symbol	Value	Use
SAD_CHECKSUM_OFF	0	Do not transmit the checksum
SAD_CHECKSUM_ON	1	Transmit the checksum

SAD500 Errors

Header: OOISADCE.h

The following are SAD500 error values that represent possible internal SAD500 problems.

Symbol	Value	Use
SAD_ERROR_LOAD_BOOT_TABLE	0x8000 (32768)	Cannot load boot table
SAD_ERROR_SLOW_MEM_FULL	0x4000 (16384)	Slow memory is full
SAD_ERROR_CREATE_BOOT_TABLE	0x2000 (8192)	Cannot create boot table
SAD_ERROR_MUST_MOVE_BLOCK	0x1000 (4096)	Must move memory block
SAD_ERROR_CAN_NOT_READ_BLOCK	0x0800 (2048)	Cannot read memory block
SAD_ERROR_CAN_NOT_FIND_MEMORY	0x0400 (1024)	Cannot find location in memory
SAD_ERROR_BAD_FORMAT	0x0200 (512)	Bad pixel mode format in memory
SAD_ERROR_CAN_NOT_WRITE_BLOCK	0x0100 (256)	Cannot write previously good block of memory
SAD_ERROR_UNKNOWN8	0x0080 (128)	Reserved error value
SAD_ERROR_UNKNOWN7	0x0040 (64)	Reserved error value
SAD_ERROR_UNKNOWN6	0x0020 (32)	Reserved error value
SAD_ERROR_UNKNOWN5	0x0010 (16)	Reserved error value
SAD_ERROR_UNKNOWN4	0x0008 (8)	Reserved error value
SAD_ERROR_UNKNOWN3	0x0004 (4)	Reserved error value

SAD_ERROR_UNKNOWN2	0x0002 (2)	Reserved error value
SAD_ERROR_UNKNOWN1	0x0001 (1)	Reserved error value
SAD_ERROR_NONE	0x0000 (0)	Reserved error value

SAD500 External Trigger Modes

Header: OOISADCE.h

Symbol	Value	Use
SAD_EXTRIG_NORMAL	0	No external trigger
SAD_EXTRIG_SOFTWARE	1	External software trigger
SAD_EXTRIG_SYNC	2	External synchronization
SAD_EXTRIG_HARDWARE	3	External hardware trigger

SAD500 Interpolation of Missing Pixels

Header: OOISADCE.h

Symbol	Value	Use
SAD_DO_NOT_INTERPOLATE_ MISSING_PIXELS	0	Do not interpolate missing pixels
SAD_INTERPOLATE_MISSING_PIXELS	1	Interpolate missing pixels

These constants apply to all of the varieties of the SAD_PIXELMODE_EVERY_N and SAD_PIXELMODE_EVERY_N_AVERAGED.

SAD500 Memory Slots

Header: OOISADCE.h

Symbol	Value	Use
SAD_SLOT_SERIALNUMBER	0	Spectrometer serial number
SAD_SLOT_WLINTERCEPT	1	Wavelength calibration intercept
SAD_SLOT_WLFIRST	2	Wavelength calibration first coefficient
SAD_SLOT_WLSECOND	3	Wavelength calibration second coefficient
SAD_SLOT_WLTHIRD	4	Wavelength calibration third coefficient
SAD_SLOT_STRAYLIGHT	5	Stray light correction factor
SAD_SLOT_NL0	6	0^{th} order linearity correction constant
SAD_SLOT_NL1	7	1^{st} order linearity correction constant
SAD_SLOT_NL2	8	2^{nd} order linearity correction constant
SAD_SLOT_NL3	9	3^{rd} order linearity correction constant
SAD_SLOT_NL4	10	4^{th} order linearity correction constant
SAD_SLOT_NL5	11	5^{th} order linearity correction constant
SAD_SLOT_NL6	12	6^{th} order linearity correction constant
SAD_SLOT_NL7	13	7^{th} order linearity correction constant
SAD_SLOT_NLORDER	14	Linearity correction order

SAD500 Memory Types

Header: OOISADCE.h

Symbol	Value	Use
SAD_MEMORY_BOTH	0	Both fast and slow memory
SAD_MEMORY_FAST	1	Fast memory only
SAD_MEMORY_SLOW	2	Slow memory only

SAD500 Pixel Modes

Header: OOISADCE.h

Symbol	Value	Use
SAD_PIXELMODE_ALL	0x0000 (0)	Transmit all pixels
SAD_PIXELMODE_EVERY_N	0x0001 (1)	Transmit every N pixels, with no boxcar averaging
SAD_PIXELMODE_EVERY_N_AVERAGED	0x0002 (2)	Transmit every N pixels, with boxcar averaging of N/2 pixels
SAD_PIXELMODE_X_TO_Y_EVERY_N	0x0003 (3)	Transmit pixels starting at pixel X to pixel Y every N pixels
SAD_PIXELMODE_SELECTED	0x0004 (4)	Transmit up to 81 selected pixels
SAD_PIXELMODE_ALL_COMPRESSED	0x0100 (256)	Transmit all pixels using compression
SAD_PIXELMODE_EVERY_N_COMPRESSED	0x0101 (257)	Transmit every N pixels, with no boxcar averaging using compression
SAD_PIXELMODE_EVERY_N_AVERAGED_COMPRESSED	0x0102 (258)	Transmit every N pixels, with boxcar averaging of N/2 pixels using compression
SAD_PIXELMODE_X_TO_Y_EVERY_N_COMPRESSED	0x0103 (259)	Transmit pixels starting at pixel X to pixel Y every N pixels using compression
SAD_PIXELMODE_SELECTED_COMPRESSED	0x0104 (260)	Transmit up to 81 selected pixels using compression
SAD_PIXELMODE_ALL_CDS	0x0200 (512)	Transmit all pixels using CDS
SAD_PIXELMODE_EVERY_N_CDS	0x0201 (513)	Transmit every N pixels, with no boxcar averaging using CDS
SAD_PIXELMODE_EVERY_N_AVERAGED_CDS	0x0202 (514)	Transmit every N pixels, with boxcar averaging of N/2 pixels using CDS
SAD_PIXELMODE_X_TO_Y_EVERY_N_CDS	0x0203 (515)	Transmit pixels starting at pixel X to pixel Y every N pixels using CDS
SAD_PIXELMODE_SELECTED_CDS	0x0204 (516)	Transmit up to 81 selected pixels using CDS
SAD_PIXELMODE_ALL_COMPRESSED_CDS	0x0300 (768)	Transmit all pixels with compression and CDS
SAD_PIXELMODE_EVERY_N_COMPRESSED_CDS	0x0301 (769)	Transmit every N pixels, with no boxcar averaging with compression and CDS
SAD_PIXELMODE_EVERY_N_AVERAGED_COMPRESSED_CDS	0x0302 (770)	Transmit every N pixels, with boxcar averaging of N/2 pixels with compression and CDS
SAD_PIXELMODE_X_TO_Y_EVERY_N_COMPRESSED_CDS	0x0303 (771)	Transmit pixels starting at pixel X to pixel Y every N pixels with compression and CDS
SAD_PIXELMODE_SELECTED_COMPRESSED_CDS	0x0304 (772)	Transmit up to 81 selected pixels with compression and CDS

SAD500 Replies

Header: OOISADCE.h

Symbol	Value	Use
SAD_ACK	6	Command was successful
SAD_NAK	21	Command was not successful
SAD_STX	2	Spectral data acquired
SAD_ETX	3	Spectral data not acquired

SAD500 Return Codes

Header: OOISADCE.h

Symbol	Value	Use
SAD_RETURN_SUCCESS	1	Command was successful
SAD_RETURN_FAILURE	0	Command was not successful
SAD_RETURN_COM_FAILURE	-1	Serial port is not initialized
SAD_RETURN_TIMEOUT	-2	Scan timed out
SAD_RETURN_COM_CAN_NOT_ FIND_COM_DLL	-3	The OOIDRV16 driver cannot find the serial port driver. If using 16-bit driver, you must have WLCC.DLL in your driver directory
SAD_RETURN_BAD_HEADER	-4	A scans data may be invalid
SAD_RETURN_BUSY_SCANNING	-5	The SAD500 is busy scanning and cannot process your command
SAD_RETURN_NOT_ALLOWED	-6	This function is not allowed
SAD_RETURN_OUT_OF_SYNC	-7	Communications became out-of-sync during data transfer
SAD_RETURN_BAD_CHECKSUM	-8	Scan checksum failed
SAD_RETURN_PARAMETER_BAD	-9999	A SAD500 function was called with a bad parameter

SAD500 Scan Modes

Header: OOISADCE.h

Symbol	Value	Use
SAD_SCAN_RETURN_IMMEDIATELY	0	Spectral data is returned immediately after the scan is complete
SAD_SCAN_TO_FAST_MEMORY	1	Spectral data is stored into fast memory
SAD_SCAN_TO_SLOW_MEMORY	2	Spectral data is stored into slow memory

SAD500 Scan Receive Status

Header: OOISADCE.h

Symbol	Value	Use
SAD_SCAN_OK	0	Scan was received by the application
SAD_SCAN_NOT_OK	1	Scan was not received by the application

SAD500 Serial Ports

Header: OOISADCE.h

Symbol	Value	Use
SAD_COM_1	1	COM1
SAD_COM_2	2	COM2
SAD_COM_3	3	COM3
SAD_COM_4	4	COM4
SAD_COM_5	5	COM5
SAD_COM_6	6	COM6
SAD_COM_7	7	COM7
SAD_COM_8	8	COM8

SAD500 Spectrometer Channels

Header: OOISADCE.h

Symbol	Value	Use
SAD_MASTER	0	Master channel
SAD_SLAVE1	1	Slave 1 channel
SAD_SLAVE2	2	Slave 2 channel
SAD_SLAVE3	3	Slave 3 channel
SAD_SLAVE4	4	Slave 4 channel
SAD_SLAVE5	5	Slave 5 channel
SAD_SLAVE6	6	Slave 6 channel
SAD_SLAVE7	7	Slave 7 channel

Standard Colors

Header: StandardColors.h

Symbol	Value (RGB)	Color Index
BLACK	0x000000	0
BLUE	0xFF0000	1
BROWN	0x004080	2
CYAN	0xFFFF00	3
DARK_BLUE	0x804000	4
DARK_GRAY	0x4E4E4E	5
DARK_GREEN	0x008000	6
GREEN	0x00FF00	7
HOT_PINK	0x8000FF	8
HUNTER_GREEN	0x408000	9
LIGHT_GRAY	0xC0C0C0	10
MAGENTA	0xFF00FF	11
MEDIUM_GRAY	0x808080	12
ORANGE	0x0080FF	13
OLIVE	0x008080	14
PINK	0xEC80FF	15
RED	0x0000FF	16
RED_BROWN	0x000080	17
YELLOW	0x00FFFF	18
WHITE	0xFFFFFF	19

These constants and index values are used by the IndexToColor and ColorToIndex functions.

Data Structures

FLOATPOINT

Header: OOIGraphCE.h

The FLOATPOINT structure is used by the OOIGraphCE_GetPointValues function to return the x and y coordinates of a specified pixel in the graph.

Datatype	Name	Use
float	x	The x value of the point
float	y	The y value of the point

PIXELMODEDATA

Header: OOISADCE.h

The PIXELMODEDATA structure is used to define the pixel transfer mode for the SAD500. The members of this structure are:

Datatype	Name	Use
WORD	pixel_mode	Specifies pixel model. Use SAD_PIXELMODE_ constants
WORD	pixel_data[81]	Pixel mode data. Specify up to 81 pixels for SAD_PIXELMODE_SELECTED
WORD	num_pixels	Number of pixels used if the pixel mode is equal to SAD_PIXELMODE_SELECTED

SADDATA

Header: OOISADCE.h

The SADDATA structure is used to return data from the SAD500. The members of this structure are:

Datatype	Name	Use
WORD	start_tag	Indicates start of SADDATA structure, which is always 0xFFFF
WORD	channel	Indicates the active channel of the spectrometer. Use SAD_MASTER, SAD_SLAVE1, etc.
WORD	scan_number	Indicates the number of the scan
WORD	scans_in_memory	Returns the number of scans remaining in memory
WORD	integration_time	Returns the integration time, in msec
WORD	integration_counter	Returns the number of integration periods which have elapsed since the last integration time change and can be used as a time stamp
WORD	pixel_mode	Indicates the pixel mode of this scan. Use SAD_PIXELMODE_ constants
WORD	pixel_data[81]	Returns extended pixel mode information if the pixel_mode does not equal SAD_PIXELMODE_ALL
WORD	scan_data[2048]	Returns the spectral data of the current scan. If a pixel mode other than SAD_PIXELMODE_ALL is being used, some of these elements are zero

REGRESSIONRESULTS

Header: OOIRegressionCE.h

The REGRESSIONRESULTS structure is used by the PerformRegression function to return the results of a linear or second-order polynomial regression

Datatype	Name	Use
float	Intercept	Returns the intercept of the regression
float	FirstCoef	Returns the first coefficient of the regression
float	SecondCoef	Returns the second coefficient of the regression. If a first order regression is selected, this number is zero
float	RSquared	Returns the R-squared of the regression, which indicates the value of the fit.

Description of Spectral Acquisition Functions

SAD_ClearMemory

Use with: SAD500 only
Header: OOISADCE.h
short SAD_ClearMemory(WORD mem)

Parameter	Use
Mem	Memory type

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an “L” to the SAD500 followed by the type of memory being cleared. The function clears all scans from fast memory, slow memory or both memories by using the appropriate SAD_MEMORY_ constants.

If you call SAD_Scan after setting the scan mode to SAD_SCAN_TO_FAST_MEMORY or SAD_SCAN_TO_SLOW_MEMORY, spectra are stored into memory rather than returned immediately. You can clear scans from memory by calling SAD_ClearMemory.

Once cleared, the scans in the memory are permanently lost.

SAD_Close

Use with: SAD500 only
Header: OOISADCE.h
short SAD_Close(WORD save)

Parameter	Use
save	SAD500 close parameter

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “C” to the SAD500 followed by the close parameter (one of the SAD_CLOSE_ constants).

The SAD_Close function allows you to store the SAD500 acquisition parameter into non-volatile memory. The next time the SAD500 is powered up, these parameters are read from memory. You can choose to store all parameters and include or exclude saving the baud rate by passing the appropriate SAD_CLOSE_ constant.

SAD_DumpFastMemory

Use with: SAD500 only
Header: OOISADCE.h
short SAD_DumpFastMemroy(void)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “D” to the SAD500, which causes the entire contents of fast memory to transfer to slow memory. If fast memory is full, this command takes several seconds.

Fast memory on the SAD500 can hold up to 15 2048-point spectra. If fast memory is full, and you try to store another spectrum, the SAD_Scan command returns a SAD_RETURN_FAILURE. You can transfer the contents of the fast memory to slow memory by calling this function.

SAD_GetADCRate

Use with: SAD500, non-functional in Palm-SPEC but follows same format
Header: OOISADCE.h
WORD SAD_GetADCRate(void)

Return Value:
This function returns the current A/D conversion frequency in kHz.

SAD500 Communication Sequence:
This function sends a “?F” to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the A/D conversion frequency in kHz.

SAD_GetADCRate is used to return the A/D conversion frequency of the SAD500. The default is 500 kHz.

SAD_GetBaudRate

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetBaudRate(void)

Return Value:
This function returns the current baud rate of the SAD500.

SAD500 Communication Sequence:
This function sends a “?K” to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the baud rate constant.

The SAD_GetBaudRate function returns one of the SAD_BAUD_ constants indicating the current baud rate.

SAD_GetBoxcarWidth

Use with: SAD500, non-functional in Palm-SPEC but follows same format
Header: OOISADCE.h
WORD SAD_GetBoxcarWidth(void)

Return Value:
This function returns the width of the pixel boxcar smoothing currently being used by the SAD500.

SAD500 Communication Sequence:
This function sends a "?B" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the pixel boxcar smoothing value.

The SAD_GetBoxcarWidth function returns the value for the pixel boxcar smoothing of the most recently completed spectral acquisition. A value of 3, for example, means that each pixel is an average of itself, 3 pixels to the left, and 3 pixels to the right.

SAD_GetChecksumMode

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetChecksumMode(void)

Return Value:
This function returns the checksum mode currently used by the SAD500.

SAD500 Communication Sequence:
This function sends a "?k" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is checksum mode of the SAD500.

The SAD_GetChecksumMode function returns a value indicating the current data checksum mode of the SAD500. For more information on SAD500 Checksum Mode, see Appendix C.

SAD_GetCompressedMode

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetCompressedMode(void)

Return Value:
This function returns the data compression mode currently used by the SAD500.

SAD500 Communication Sequence:
This function sends a "?G" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the state of the data compression of the SAD500.

The SAD_GetCompressedMode function returns a value indicating the current data compression mode of the SAD500. For more information on SAD500 Data Compression, see Appendix B.

SAD_GetError

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetError(void)

Return Value:
This function returns the current error status of the SAD500.

SAD500 Communication Sequence:
This function sends a "?q" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is its the error status.

During normal acquisition, it is possible for a SAD500 to have an internal error. When this occurs, you can call the SAD_GetError function to retrieve and clear the internal error.

SAD_GetExternalTrigger

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetExternalTrigger(void)

Return Value:
This function returns the current external trigger mode of the SAD500.

SAD500 Communication Sequence:
This function sends a "?T" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the external trigger mode.

You can configure any Ocean Optics spectrometer to initiate a spectral acquisition in response to an external event. The SAD_GetExternalTrigger function returns one of the SAD_EXTRIG_ constants indicating the external trigger mode of the most recently completed spectral acquisition.

SAD_GetFastMemoryAvailable

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetFastMemoryAvailable(void)

Return Value:
This function returns the amount of scans that can be stored in fast memory.

SAD500 Communication Sequence:
This function sends a "X" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the number of scans that can be stored in fast memory.

The SAD500 has two types of memory: fast memory and slow memory. You can write to fast memory in a few microseconds, but the data is lost if the SAD500 loses power. Writing to slow memory takes tens or hundreds of milliseconds, but the data is retained when the SAD500 loses power. The SAD_GetFastMemoryAvailable returns the number of spectra that can be stored in fast memory.

SAD_GetFullPixelMode

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetFullPixelMode(PIXELMODEDATA* pix)

Parameter	Use
pix	A pointer to a PIXELMODEDATA structure that is filled

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a "?p" to the SAD500. After the SAD500 acknowledges the command, the current pixel mode information is returned by the SAD500. The number of WORDs returned depends on the current pixel mode.

Transferring a complete 2048-point spectrum over the RS-232 serial port takes some time. At 115200 baud, this uncompressed transfer takes about 300 milliseconds. If a particular application does not require a 2048-point spectrum, you can configure the SAD500 to transfer a subset of the complete spectral data. The SAD_GetFullPixelMode function fills a PIXELMODEDATA structure with the current SAD500 pixel mode.

SAD_GetIntegrationCounter

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetIntegrationCounter(void)

Return Value:
This function returns the integration counter.

SAD500 Communication Sequence:
This function sends a "t" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the number of integration periods that have elapsed since the SAD500 was powered on or reset.

The SAD500 has an internal counter that keeps track of how many integration cycles have been completed. The SAD_GetIntegrationCounter function returns this counter, and this value can be used to time stamp a spectral acquisition with a resolution equal to the integration time.

SAD_GetIntegrationTime

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetIntegraitonTime(void)

Return Value:
This function returns the current integration time of the SAD500.

SAD500 Communication Sequence:
This function sends a "?I" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the integration time in msec.

The integration time is the length of time that the CCD detector acquires data. The SAD_GetIntegrationTime returns the integration time (in msec) of the most recently completed spectral acquisition.

SAD_GetInterpolateMissingPixels

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetInterpolateMissingPixels(void)

Return Value:
This function returns the 1 if the driver is to interpolate missing pixels when in the following pixel modes SAD_PIXELMODE_EVERY_N, SAD_PIXELMODE_EVERY_N_AVERAGED, and SAD_PIXELMODE_X_TO_Y_EVERY_N.

SAD500 Communication Sequence:
This function does not communicate directly with the SAD500 as the interpolation occurs within the device driver, not the SAD500.

In pixel modes SAD_PIXELMODE_EVERY_N, SAD_PIXELMODE_EVERY_N_AVERAGED and SAD_PIXELMODE_X_TO_Y_EVERY_N, and their compressed and CDS variants, the SAD500 transfers a subset of the complete 2048-point spectrum. You can instruct the driver to compute the interpolated intensity for pixels that are not transferred. The SAD_GetInterpolatedMissingPixels returns the present state of this optional interpolation.

SAD_GetNumberOfScansInMemory

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetNumberOfScansInMemory(WORD mem)

Parameter	Use
mem	Memory type

Return Value:
This function returns the number of scans in the memory of the SAD500.

SAD500 Communication Sequence:
This function sends a “W” to the SAD500 followed by the mem parameter. After the SAD500 receives the command, the next WORD returned from the SAD500 is the number of scans in memory.

The SAD500 has two types of memory: fast memory and slow memory. It is possible to write data to fast memory in a few microseconds, but the data is lost if the SAD500 loses power. Writing to slow memory takes tens or hundreds of milliseconds, but data is saved if the SAD500 loses power. The SAD_GetNumberOfScansInMemory function returns the number of scans in fast or slow memory.

SAD_GetNumberOfScansToStore

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetNumberOfScansToStore(void)

Return Value:
This function returns the number of scans that are stored in the next spectral acquisition.

SAD500 Communication Sequence:
This function sends a “?N” to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the number of scans to store.

You can instruct the SAD500 to perform a number of spectral acquisitions and to store the result in memory rather than return over the serial port. The SAD_GetNumberOfScansToStore function returns the number of scans that are stored in the next spectral acquisition.

SAD_GetPixelMode

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetPixelMode(void)

Return Value:
This function returns the current pixel mode of the SAD500.

SAD500 Communication Sequence:
This function sends a "?P" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the current pixel mode.

Transferring a complete 2048-point spectrum over the RS-232 serial port takes some time. At 115200 baud, this uncompressed transfer takes about 300 milliseconds. If a particular application does not require a 2048-point spectrum, the SAD500 can transfer a subset of the complete spectral data. The SAD_GetPixelMode function returns one of the SAD_PIXELMODE_ constants indicating the pixel mode of the next spectral acquisition.

SAD_GetS1024DWCDSMode

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetS1024DWCDSMode(void)

Return Value:
This function returns the correlated double sampling (CDS) mode currently being used by the SAD500.

SAD500 Communication Sequence:
This function sends a "?h" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 CDS mode. Correlated double sampling is used only with the S1024DW. For more information on Correlated Double Sampling, see Appendix C.

SAD_GetScanMode

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetScanMode(void)

Return Value:
This function returns the current scan mode of the SAD500.

SAD500 Communication Sequence:
This function sends a "?M" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the current scan mode.

You can configure the SAD500 to return spectra immediately after they are acquired or to store spectra in memory. The SAD_GetScanMode returns one of the SAD_SCAN_ constants indicating the current scan mode.

SAD_GetScansToAdd

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetScansToAdd(void)

Return Value:
This function returns the current number of scans to add for the SAD500.

SAD500 Communication Sequence:
This function sends a "?A" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the number of scans to add.

You can configure the SAD500 to add up to 15 discrete spectral acquisitions. This is equivalent to averaging up to 15 spectral acquisitions. If you wish to average more than one spectral acquisition, this feature reduces the amount of time necessary to transfer spectral data, since fewer spectra have to transmit over the serial interface. The SAD_GetScansToAdd function returns the number of scans added in the next spectral acquisition.

SAD_GetSlowMemoryAvailable

Use with: SAD500 only
Header: OOISADCE.h
WORD SAD_GetSlowMemoryAvailable(void)

Return Value:
This function returns the number of kilobytes of slow memory available in the SAD500.

SAD500 Communication Sequence:
This function sends a "U" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the number of free kilobytes of slow memory.

The SAD500 has two types of memory: fast memory and slow memory. You can write to fast memory in a few microseconds, but the data is lost if the SAD500 loses power. Writing to slow memory takes tens or hundreds of milliseconds, but the data is retained when the SAD500 loses power. The SAD_GetSlowMemoryAvailable returns the number of kilobytes of free slow memory.

SAD_GetSpectrometerChannel

Use with: SAD500, non-functional in Palm-SPEC but follows same format
Header: OOISADCE.h
WORD SAD_GetSpectrometerChannel(void)

Return Value:
This function returns the current spectrometer channel selected in the SAD500.

SAD500 Communication Sequence:
This function sends a "?H" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the current spectrometer channel.

You can attach up to 8 S2000 spectrometer channels (master plus 7 slaves) to the SAD500. The SAD_GetSpectrometerChannel function returns the spectrometer channel acquiring data during the next spectral acquisition.

SAD_GetStrobeEnable

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetStrobeEnable(void)

Return Value:
This function returns the status of the strobe enable in the SAD500.

SAD500 Communication Sequence:
This function sends a "?J" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the status of the strobe enable.

You can connect an external pulsed lamp (e.g., the Ocean Optics PX-2 Pulsed Xenon Light Source) to an S2000 or S1024DW spectrometer. In this configuration, the spectrometer controls the firing of the lamp. The SAD_GetStrobeEnable function returns 1 if the external strobe control is enabled, 0 if it is disabled.

SAD_GetStoredFloat

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_GetStoredFloat(WORD slot,float* value)

Parameter	Use
slot	Desired memory slot, use SAD_SLOT_ constant
value	Pointer to the floating-point variable to receive the value

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "?x" to the SAD500 followed by the slot parameter. After the SAD500 acknowledges the command, a string is returned by the SAD500 containing the value of the stored parameter. The driver then converts this string into a floating-point value.

SAD_GetStoredInt

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_GetStoredInt(WORD slot,int* value)

Parameter	Use
slot	Desired memory slot, use SAD_SLOT_ constant
value	Pointer to the integer variable to receive the value

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "?x" to the SAD500 followed by the slot parameter. After the SAD500 acknowledges the command, a string is returned by the SAD500 containing the value of the stored parameter. The driver then converts this string into an integer value.

SAD_GetStoredString

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_GetStoredString(WORD slot,char* value)

Parameter	Use
slot	Desired memory slot, use SAD_SLOT_ constant
value	Pointer to the string variable to receive the value. You must ensure that this buffer is at least 16 characters long

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "?x" to the SAD500 followed by the slot parameter. After the SAD500 acknowledges the command, a string is returned by the SAD500 containing the value of the stored parameter.

SAD_GetVersion

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
WORD SAD_GetVersion(void)

Return Value:
This function returns the version of the code in the microcontroller of the SAD500.

SAD500 Communication Sequence:
This function sends a "v" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the version of the code in the microcontroller. A value of 1020 is interpreted as 1.02.0.

SAD_Init

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_Init(short com, short baud)

Parameter	Use
com	Serial port identifier
baud	Baud rate constant, use SAD_BAUD_

Return Value:
This function returns one of the SAD_COM_ constants upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function initializes the driver and potentially changes the baud rate of the SAD500.

The SAD_Init function opens the serial port for communication with the SAD500 at the specified baud rate.

SAD_IsPalmSpec

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
BOOL SAD_IsPalmSpec(void)

Return Value:
This function returns TRUE if the spectrometer connected to the computer is a Palm-SPEC, FALSE if it is an S2000/SAD500.

SAD500 Communication Sequence:
This function sends a = to the SAD500. If the A/D returns an ACK, the spectrometer is a Palm-SPEC.

The SAD_IsPalmSpec function is used to differentiate between a Palm-SPEC and an S2000/SAD500 spectrometer.

SAD_ReadoutOneScan

Use with: SAD500 only
Header: OOISADCE.h
short SAD_ReadoutOneScan(WORD mem, SADDATA* sad)

Parameter	Use
mem	Type of memory
sad	Pointer to a SADDATA structure

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “Z” to the SAD500 followed by the memory parameter (a SAD_MEMORY_ constant). After the SAD500 acknowledges the command, it sends one complete scan.

The SAD_ReadoutOneScan function reads one spectrum from either fast memory (mem = SAD_MEMORY_FAST) or slow memory (mem = SAD_MEMORY_SLOW). If a scan is not present in the selected memory, the SAD_ReadoutOneScan returns SAD_RETURN_FAILURE.

SAD_Reset

Use with: SAD500 only
Header: OOISADCE.h
short SAD_Reset(void)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “Q” to the SAD500. After the SAD500 acknowledges the command, the SAD500 resets.

The SAD_Reset function resets the SAD500, but does not change the baud rate. It is equivalent to powering off and then powering on the SAD500. All scans in fast memory are lost, as are all current acquisition parameters.

Once the SAD500 is reset, all scans in fast memory and all acquisition parameters are permanently lost.

SAD_Scan

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_Scan(SADDATA* sad)

Parameter	Use
sad	Pointer to a SADDATA structure

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “S” to the SAD500. After the SAD500 acknowledges the command, a spectral acquisition is performed using the acquisition parameters that have been previously passed to the SAD500.

The SAD_Scan function instructs the SAD500 to perform a spectral acquisition. All acquisition parameters (integration time, scans to add, etc.) must have been sent to the SAD500 prior to the call to SAD_Scan. Once a spectral acquisition is initiated, the SAD500 is completely unresponsive to any command.

SAD_ScanWithAverage

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_ScanWithAverage(SADDATA* sad,short average,float* data)

Parameter	Use
sad	Pointer to a SADDATA structure
average	Number of scans to average
data	Pointer to a 2048 element array for return of spectral data

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends several commands to the SAD500. Depending on the number of scans being acquired, one or several commands to SAD_SetScansToAdd and SAD_Scan are made.

The SAD_ScanWithAverage function instructs the SAD500 to perform a spectral acquisition. All acquisition parameters (integration time, smoothing, etc.) must have been sent to the SAD500 prior to the call to SAD_ScanWithAverage. Once a spectral acquisition is initiated, the SAD500 is completely unresponsive to any command.

SAD_ScanReceivedOK

Use with: SAD500 only
Header: OOISADCE.h
short SAD_ScanReveivedOK(WORD ok)

Parameter	Use
ok	Scan received ok (use SAD_SCAN_ constants)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "O" to the SAD500 followed by the ok parameter. After the SAD500 acknowledges the command, the SAD500 retransmits the spectrum if the ok parameter was 1.

The SAD_ScanReceivedOK function informs the SAD500 if a scan was received with no errors. Call this function after each SAD_ReadoutOneScan to let the SAD500 know that the scan was received without error. If a value of 1 is used when calling this function, the SAD500 retransmits the spectrum.

SAD_SetADCRate

Use with: SAD500, non-functional in Palm-SPEC but follows same format
Header: OOISADCE.h
short SAD_SetADCRate(WORD rate)

Parameter	Use
rate	The A/D conversion frequency in kHz

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "F" to the SAD500 followed by the rate parameter. After the SAD500 acknowledges the command, the A/D conversion frequency is changed.

The SAD_SetADCRate is used to change the A/D conversion frequency. The default A/D conversion is 500kHz. The maximum A/D conversion frequency is also 500kHz.

SAD_SetBaudRate

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetBaudRate(WORD baud)

Parameter	Use
baud	The baud rate constant (use SAD_BAUD_ constants)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “K” to the SAD500 followed by the baud parameter. The SAD500 acknowledges the command, and changes its communication baud rate. The driver then waits 50 msec, and changes the computer’s baud rate. The driver then resends the “K” to the SAD500 followed by the baud parameter. The SAD500 then acknowledges the command.

The SAD_SetBaudRate changes the speed of the communication between the computer and the SAD500.

SAD_SetBoxcarWidth

Use with: SAD500, non-functional in Palm-SPEC but follows same format
Header: OOISADCE.h
short SAD_SetBoxcarWidth(WORD box)

Parameter	Use
box	The pixel boxcar smoothing value

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “B” to the SAD500 followed by the desired pixel smoothing width. After the SAD500 acknowledges the command, the boxcar smoothing width is changed.

There are two ways that the SAD500 can average data: average across time or average across the spectrum. The SAD_SetBoxcarWidth sets the SAD500 averaging across the spectrum. A value of 5, for example, averages each data point with itself, 5 points to its left and 5 points to its right.

SAD_SetChecksumMode

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetChecksumMode(WORD mode)

Parameter	Use
Mode	The SAD500 checksum mode (use SAD_CHECKSUM_ constants)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an “h” to the SAD500 followed by the rate parameter. After the SAD500 acknowledges the command, checksum mode is changed.

The SAD_SetChecksumMode allows the SAD500 to transfer a 16-bit unsigned checksum for the spectral data. This checksum is transferred immediately after the 0xfffd at the end of the spectral data. For more information on the SAD500 Checksum Calculation, see Appendix D.

SAD_SetCompressedMode

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetCompressedMode(WORD mode)

Parameter	Use
Mode	The SAD500 compression mode (use the SAD_COMPRESSION_ constants)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a "G" to the SAD500 followed by the rate parameter. After the SAD500 acknowledges the command, the compression mode is changed.

The SAD_SetCompressedMode function allows the SAD500 to transmit compressed spectral data to the computer. This data compression allows for faster data transfer rates. For more information on SAD500 Data Compression, see Appendix B.

SAD_SetExternalTrigger

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetExternalTrigger(WORD trig)

Parameter	Use
trig	The external trigger constant (use the SAD_EXTRIG_ constants)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a "T" to the SAD500 followed by the desired external trigger mode (one of the SAD_EXTRIG_ constants). After the SAD500 acknowledges the command, the external trigger mode is changed.

In the default mode, the SAD500 acquires a spectrum immediately after a call to the SAD_Scan function. If desired, an external event can trigger the acquisition. The SAD_SetExternalTrigger function allows you to instruct the SAD500 to acquire a spectrum, but only after it receives an external trigger.

SAD_SetIntegrationTime

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetIntegrationTime(WORD msec)

Parameter	Use
msec	The desired integration time in msec

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "I" to the SAD500 followed the desired integration time in msec. After the SAD500 acknowledges the command, the integration time is changed.

The integration time of the spectrometer is roughly equivalent to the shutter speed of a camera. The integration time determines how long the CCD pixels accumulate light before the spectrum is considered done. The SAD_SetIntegrationTime function is used to change the integration time of the SAD500.

SAD_SetInterpolateMissingPixels

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetInterpolateMissingPixels(WORD enabled)

Parameter	Use
enabled	Interpolation method (use the SAD_INTERPOLATE_MISSING_PIXELS_ constants)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
There is no communication to the SAD500 for this command.

In pixel modes SAD_PIXELMODE_EVERY_N, SAD_PIXELMODE_EVERY_N_AVERAGED or SAD_PIXELMODE_X_TO_Y_EVERY_N, and their compressed and correlated doubled sampled variants, the SAD500 transfers a subset of the complete 2048-point spectrum. You can instruct the driver to compute the interpolated intensity for pixels that are not transferred. The SAD_SetInterpolatedMissingPixels sets the state of this optional interpolation.

SAD_SetNumberOfScansToStore

Use with: SAD500 only
Header: OOISADCE.h
short SAD_SetNumberOfScansToStore(WORD add)

Parameter	Use
add	The number of scans to store

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "N" to the SAD500 followed by the number of scans to store. After the SAD500 acknowledges the command, the number of scans to add applies to all subsequent spectral acquisitions.

You can instruct the SAD500 to perform a number of spectral acquisitions and store the results in memory rather than return them over the serial port. The SAD_GetNumberOfScansToStore function returns the number of scans stored in the next spectral acquisition.

SAD_SetPalmSpecTimer

Use with: Palm-SPEC only
Header: OOISADCE.h
short SAD_SetPalmSpecTimer(void)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “y” to the Palm-SPEC. After the Palm-SPEC acknowledges the command, an internal 16-bit timer is allocated for the adjustment of the integration time.

It is necessary to use this command after calling the SAD_Init function if you're using a Palm-SPEC.

SAD_SetPixelMode

Use with: SAD500, non-functional in Palm-SPEC but follows same format
Header: OOISADCE.h
short SAD_SetPixelMode(PIXELMODEDATA* pix)

Parameter	Use
pix	Pointer to a PIXELMODEDATA structure

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “P” to the SAD500 followed by the pixel mode information. After the SAD500 acknowledges the command, the pixel mode of the SAD500 is changed.

Transferring a complete 2048-point spectrum over the RS-232 serial port takes time. At 115200 baud, this uncompressed transfer takes ~300 milliseconds. If a particular application does not require a 2048-point spectrum, you can configure the SAD500 to transfer a subset of the data. The SAD_SetPixelMode function configures the SAD500 and the driver to the desired pixel mode.

SAD_SetS1024DWCDSMode

Use with: SAD500 only
Header: OOISADCE.h
short SAD_SetS1024DWCDSMode(WORD mode)

Parameter	Use
mode	Correlated double sampling mode (use the SAD_CDS_ constants)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
There is no communication to the SAD500 for this command.

This function sends an “h” to the SAD500 followed by the desired correlated double sampling mode. After the SAD500 gets the command, the CDS mode of the SAD500 changes. CDS is used only with the S1024DW. For more information on Correlated Double Sampling, see Appendix C.

SAD_SetScanMode

Use with: SAD500 only
Header: OOISADCE.h
short SAD_SetScanMode(WORD mode)

Parameter	Use
mode	The desired scan mode (use the SAD_SCAN_ constants

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "M" to the SAD500 followed by the desired scan mode. After the SAD500 acknowledges the command, the scan mode of the SAD500 is changed.

You can configure the SAD500 to return spectra immediately after they are acquired or to store them in memory. The SAD_SetScanMode sets the SAD500 to a new scan mode.

SAD_SetScansToAdd

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetScansToAdd(WORD add)

Parameter	Use
add	The number of scans to add

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an "A" to the SAD500 followed by the desired number of scans. After the SAD500 acknowledges the command, the SAD500 acquires the specified number of scans with each call to SAD_Scan.

You can configure the SAD500 to add up to 15 discrete spectral acquisitions. This is equivalent to averaging up to 15 spectral acquisitions. If you wish to average more than one spectral acquisition, this feature reduces the amount of time necessary to transfer spectral data, since fewer spectra are transmitted over the serial interface. The SAD_SetScansToAdd function sets the number of scans added in the next spectral acquisition.

SAD_SetSerialPort

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetSerialPort(WORD port)

Parameter	Use
port	Serial port identifier (use the SAD_COM constants)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function does not communicate directly with the SAD500.

The driver communicates with the SAD500 via a serial port. When the serial port is initialized, pass the parameter indicating the serial port number to the SAD_Init function. If, at a later time you wish to communicate via a different serial port, you must call the SAD_SetSerialPort function.

SAD_SetSpectrometerChannel

Use with: SAD500, non-functional in Palm-SPEC but follows same format
Header: OOISADCE.h
short SAD_SetSpectrometerChannel(WORD chan)

Parameter	Use
chan	The desired spectrometer channel (use SAD_MASTER, SAD_SLAVE1, etc.)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “H” to the SAD500 followed by the desired spectrometer channel. After the SAD500 acknowledges the command, the channel for the next acquisition is set.

You can attached up to 8 S2000 or S1024DW spectrometer channels (master plus 7 slaves) to the SAD500. The SAD_SetSpectrometerChannel function sets the spectrometer channel that is acquired during the next spectral acquisition.

SAD_SetStrobeEnable

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetStrobeEnable (WORD ena)

Parameter	Use
ena	Strobe enable value (1 enables the strobe, 0 disables the strobe)

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “J” to the SAD500 followed by the desired strobe enable state. After the SAD500 acknowledges the command, the strobe enable status is immediately changed.

You can connect an external pulsed lamp (e.g., the Ocean Optics PX-2 Pulsed Xenon Light Source) to the S2000 and S1024DW spectrometers. In this configuration, the spectrometer controls the firing of the lamp. The SAD_SetStrobeEnable function sets the state of the strobe enable.

SAD_SetStoredFloat

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetStoredFloat(WORD slot,float value)

Parameter	Use
slot	Desired memory slot, use SAD_SLOT_ constant
value	The floating-point value stored

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “x” to the SAD500 followed slot parameter. Then, a 15-character string representing the floating-point value stored is sent to the SAD500. After the SAD500 acknowledges the command, the value is stored.

SAD_SetStoredInt

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetStoredInt(WORD slot,int value)

Parameter	Use
slot	Desired memory slot, use SAD_SLOT_ constant
value	The integer value stored

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “x” to the SAD500 followed slot parameter. Then, a 15-character string representing the integer value stored is sent to the SAD500. After the SAD500 acknowledges the command, the value is stored.

SAD_SetStoredString

Use with: SAD500 and Palm-SPEC
Header: OOISADCE.h
short SAD_SetStoredString(WORD slot,char* value)

Parameter	Use
slot	Desired memory slot, use SAD_SLOT_ constant
value	A pointer to the string stored

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “x” to the SAD500 followed slot parameter. Then, the string (up to 15 characters) stored is sent to the SAD500. After the SAD500 acknowledges the command, the value is stored.

Description of Regression Function

PerformRegression

Header: OOIRegressionCE.h
void PerformRegression(float* x, float* y, int values, int order, REGRESSIONRESULTS* reg)

Parameter	Use
x	A pointer to an array containing the x values (independent values) for the regression
y	A pointer to an array containing the y values (dependent values) for the regression
values	The number of data points in the set
order	The order of the regression – use 1 for a first order, 2 for a second order
reg	A pointer to a REGRESSIONRESULTS structure to receive the results of the regression

Return value:
This function has no return value.

This function performs either a first-order linear or a second-order polynomial regression (using the least-squares method) on the data set defined by the x and y variable. The results of this regression are placed in a REGRESSIONRESULT structure, and include the intercept, first coefficient, second coefficient (if a second-order regression is specified) and the R^2 value.

Description of Graphics Functions

OOIGraphCE_AutoscaleAxis

Header: OOIGraphCE.h
void OOIGraphCE_AutoscaleAxis(int axis, BOOL autoscale)

Parameter	Use
axis	The axis autoscaled. Valid values are X_AXIS and Y_AXIS
autoscale	TRUE if the axis is automatically scaled on the next redraw, otherwise FALSE

Return value:
This function has no return value.

The OOIGraphCE_AutoscaleAxis function is used to enable or disable autoscaling of either the X- or Y- axis of the graph. If autoscaling an axis is enabled before the graph is rendered, the OOIGraphCE library finds the minimum and maximum values of the dataset, and automatically adjusts the range of the axis to precisely contain the minimum and maximum values. You must use the OOIGraphCE_GetXMin, OOIGraphCE_GetXMax, OOIGraphCE_GetYMin and OOIGraphCE_GetYMax functions to retrieve the ranges of the X and Y axes.

OOIGraphCE_Close

Header: OOIGraphCE.h
void OOIGraphCE_Close(void)

Parameters:
This function requires no parameters.

Return value:
This function has no return value.

The OOIGraphCE_Close function closes the graphics library and frees any allocated memory.

Failure to call this function after you are finished using the OOIGraphCE library results in memory leaks.

OOIGraphCE_GetBackgroundColor

Header: OOIGraphCE.h
COLORREF OOIGraphCE_GetBackgroundColor(void)

Return value:
The RGB color of the background of the graph.

The OOIGraphCE_GetBackgroundColor returns the RGB color of the background of the graph. The StandardColors.h header contains the RGB values of several standard colors. Set the color of the background of the graph with the OOIGraphCE_SetBackgroundColor function.

OOIGraphCE_GetCursorColor

Header: OOIGraphCE.h
COLORREF OOIGraphCE_GetCursorColor(void)

Return value:
The RGB color of the cursor in the graph.

The OOIGraphCE_GetCursorColor returns the RGB color of the cursor. The StandardColors.h header contains the RGB values of several standard colors. Set the color of the cursor in the graph with the OOIGraphCE_SetCursorColor function.

OOIGraphCE_GetCursorPosition

Header: OOIGraphCE.h
int OOIGraphCE_GetCursorPosition(void)

Parameters:
This function takes no parameters.

Return value:
The 0-based index of the X-dataset indicating the current cursor position.

The OOIGraphCE_GetCursorPosition function retrieves the current position of the cursor. The returned value is the 0-based position of the cursor in the X-dataset.

OOIGraphCE_GetCursorXValue

Header: OOIGraphCE.h
float OOIGraphCE_GetCursorXValue(void)

Parameters:
This function takes no parameters.

Return value:
The X-axis value at the position of the cursor's position.

The OOIGraphCE_GetCursorXValue function returns the X-axis value of the cursor location. Use the OOIGraphCE_GetCursorYValue function to retrieve the Y-axis value.

OOIGraphCE_GetCursorYValue

Header: OOIGraphCE.h
float OOIGraphCE_GetCursorYValue(void)

Parameters:
This function takes no parameters.

Return value:
The Y-axis value at the position of the cursor's position.

The OOIGraphCE_GetCursorYValue function returns the Y-axis value of the cursor location. Use the OOIGraphCE_GetCursorXValue function to retrieve the X-axis value.

OOIGraphCE_GetFrameColor

Header: OOIGraphCE.h
COLORREF OOIGraphCE_GetFrameColor(void)

Return value:
The RGB color of the frame around the graph.

The OOIGraphCE_GetFrameColor returns the RGB color of the frame around the graph. The StandardColors.h header contains the RGB values of several standard colors. Set the color of the frame around the graph with the OOIGraphCE_SetFrameColor function.

OOIGraphCE_GetGridColor

Header: OOIGraphCE.h
COLORREF OOIGraphCE_GetGridColor(void)

Return value:
The RGB color of the grid in the graph.

The OOIGraphCE_GetGridColor returns the RGB color of the grid in the graph. The StandardColors.h header contains the RGB values of several standard colors. Set the color of the grid in the graph with the OOIGraphCE_SetGridColor function.

OOIGraphCE_GetPointValues

Header: OOIGraphCE.h
FLOATPOINT OOIGraphCE_GetPointValues(POINT p)

Parameter	Use
p	A POINT structure identifying the window coordinates of the point

Return value:
This function returns a FLOATPOINT structure containing the graph coordinates of the point specified in the p parameter.

The OOIGraphCE_GetPointValues function returns the X and Y values of a point (p) within the graph. The p is specified relative to the main window of the application. To determine if a specified point is within the graph area prior to calling this function, use the OOIGraphCE_IsPointInGraph function. If the point is not within the area of the graph, the values returned in the FLOATPOINT structure reflect the point in the graph closest to the point specified. For example, if you specify a point to the right of the active area of the graph, the x member of the FLOATPOINT structure contains the maximum value displayed on the X-axis.

OOIGraphCE_GetTraceColor

Header: OOIGraphCE.h
COLORREF OOIGraphCE_GetTraceColor(void)

Return value:
The RGB color of the trace in the graph.

The OOIGraphCE_GetTraceColor returns the RGB color of the trace in the graph. The StandardColors.h header contains the RGB values of several standard colors. Set the color of the trace with the OOIGraphCE_SetTraceColor function.

OOIGraphCE_GetXMax

Header: OOIGraphCE.h
float OOIGraphCE_GetXMax(void)

Return value:
The maximum value displayed on the X-axis.

The OOIGraphCE_GetXMax function returns the maximum value displayed on the X-axis. Set the range of the X-axis set using the OOIGraphCE_SetXRange function.

OOIGraphCE_GetXMin

Header: OOIGraphCE.h
float OOIGraphCE_GetXMin(void)

Return value:
The minimum value displayed on the X-axis.

The OOIGraphCE_GetXMin function returns the minimum value displayed on the X-axis. Set the range of the X-axis using the OOIGraphCE_SetXRange function.

OOIGraphCE_GetXTicks

Header: OOIGraphCE.h
int OOIGraphCE_GetXTicks(void)

Return value:
The number of tick marks on the X-axis.

The OOIGraphCE_GetXTicks returns the number of tick marks along the X-axis. The number of tick marks along the X-axis is set by using the OOIGraphCE_SetXTicks.

OOIGraphCE_GetYMax

Header: OOIGraphCE.h
float OOIGraphCE_GetYMax(void)

Return value:
The maximum value displayed on the Y-axis.

The OOIGraphCE_GetYMax function returns the maximum value displayed on the Y-axis. Set the range of the Y-axis using the OOIGraphCE_SetYRange function.

OOIGraphCE_GetYMin

Header: OOIGraphCE.h
float OOIGraphCE_GetYMin(void)

Return value:
The minimum value displayed on the Y-axis.

The OOIGraphCE_GetYMin function returns the minimum value displayed on the Y-axis. Set the range of the Y-axis using the OOIGraphCE_SetYRange function.

OOIGraphCE_GetYTicks

Header: OOIGraphCE.h
int OOIGraphCE_GetYTicks(void)

Return value:
The number of tick marks on the Y-axis.

The OOIGraphCE_GetYTicks returns the number of tick marks along the Y-axis. The number of tick marks along the Y-axis is set by using the OOIGraphCE_SetYTicks.

OOIGraphCE_Init

Header: OOIGraphCE.h
void OOIGraphCE_Init(void)

Parameters:
This function requires no parameters.

Return value:
This function has no return value.

The OOIGraphCE_Init function initializes the graphics library, allocates memory for the storage of data, and prepares to draw the graph. By default, the graphics library draws data consisting of 2048 pairs of floating point values. For more or less than 2048 points, you must redefine the constant PIXELS contained in the OOIGraphCE.h library, and then call the OOIGraphCE_Init function.

Failure to call this function before calling any other OOIGraphCE functions results in a critical failure.

OOIGraphCE_IsPointInGraph

Header: OOIGraphCE.h
int OOIGraphCE_IsPointInGraph(int x, int y)

Parameter	Use
x	X-coordinate, relative to the application's window
y	Y-coordinate, relative to the application's window

Return value:
XANDY if the point is totally within the area of the graph; XONLY if it is within the horizontal area of the graph, but not the vertical area; YONLY if it is not within the horizontal area of the graph, but within the vertical area; or NEITHER if the point is entirely out of the extents of the graph.

The OOIGraphCE_IsPointInGraph function is used to determine if a point (specified in coordinates relative to the application's main window) is within in the active area of the graph.

OOIGraphCE_Redraw

Header: OOIGraphCE.h
void OOIGraphCE_Redraw(HWND hWnd, BOOL DrawFrame)

Parameter	Use
hWnd	Specifies the Windows handle of the window in which the graph is drawn
DrawFrame	Indicates if the frame of the graph is redrawn

Return value:
This function has no return value.

The OOIGraphCE_Redraw function redraws the graph in the specified window. The output of the graph is not directly associated with any specific window until this function is called. This flexibility allows you to easily redraw the same graph in multiple windows by simply calling this function multiple times and passing a different hWnd parameter with each call. If you want the graph redrawn in an area of a given window, it is recommended that you place a static frame in the desired area, and passing the HWND parameter of the static frame to this function.

OOIGraphCE_SetBackgroundColor

Header: OOIGraphCE.h
void OOIGraphCE_SetBackgroundColor(COLORREF color)

Parameter	Use
color	The desired graph background color. Common color names and their RGB values are located in the StandardColors.h header

Return value:
This function has no return value.

The OOIGraphCE_SetBackgroundColor function sets the color of the background of the graph area.

OOIGraphCE_SetCursorColor

Header: OOIGraphCE.h
void OOIGraphCE_SetCursorColor(COLORREF color)

Parameter	Use
color	The desired graph cursor color. Common color names and their RGB values are located in the StandardColors.h header

Return value:
This function has no return value.

The OOIGraphCE_SetCursorColor function sets the color of the cursor, and affects both the horizontal and vertical cursors.

OOIGraphCE_SetCursorPosition

Header: OOIGraphCE.h
void OOIGraphCE_SetCursorPosition(int point)

Parameter	Use
point	The x-axis data set position of the cursor

Return value:
This function has no return value.

The OOIGraphCE_SetCursorPostition function sets the position of the cursor. The position is a 0-based index of the X-axis data set. Use the OOIGraphCE_GetCursorXValue and OOIGraphCE_GetCursorYValue to retrieve the X and Y values of the current dataset at the specified cursor location.

OOIGraphCE_SetFrameColor

Header: OOIGraphCE.h
void OOIGraphCE_SetFrameColor(COLORREF color)

Parameter	Use
color	The desired graph frame color. Common color names and their RGB values are located in the StandardColors.h header

Return value:
This function has no return value.

The OOIGraphCE_SetFrameColor sets the color of a frame around the graph area. Control the size of this frame by altering the top, bottom, left and right variables located in the OOIGraphCE_Init function.

OOIGraphCE_SetGridColor

Header: OOIGraphCE.h
void OOIGraphCE_SetGridColor(COLORREF color)

Parameter	Use
color	The desired graph grid color. Common color names and their RGB values are located in the StandardColors.h header

Return value:
This function has no return value.

The OOIGraphCE_SetGridColor function sets the color of the grid in the graph. Both the horizontal the vertical grids are affected.

OOIGraphCE_SetGridLineType

Header: OOIGraphCE.h
void OOIGraphCE_SetGridLineType(int type)

Parameter	Use
type	The style of the grid lines. Use the SOLID_LINE or DASHED_LINE constants

Return value:
This function has no return value.

The OOIGraphCE_SetGridLineType sets the style of the grid lines drawn on the graph. To toggle the display of the grid, use the OOIGraphCE_ShowGrid function. To set the frequency of the tick marks, use the OOIGraphCE_SetXTicks and OOIGraphCE_SetYTicks functions.

OOIGraphCE_SetNumValues

Header: OOIGraphCE.h
void OOIGraphCE_SetNumValues(int values)

Parameter	Use
values	The number of x-y pairs in the dataset

Return value:
This function has no return value.

The OOIGraphCE_SetNumValues sets the number of x-y pairs drawn each time the graph is rendered. By default, this value is 2048, corresponding to the number of pixels in the S2000 and Palm-SPEC spectrometers.

OOIGraphCE_SetTraceColor

Header: OOIGraphCE.h
void OOIGraphCE_SetTraceColor(COLORREF color)

Parameter	Use
color	The desired graph trace color. Common color names and their RGB values are located in the StandardColors.h header

Return value:
This function has no return value.

The OOIGraphCE_SetTraceColor function sets the color of the trace for the graph.

OOIGraphCE_SetTraceType

Header: OOIGraphCE.h
void OOIGraphCE_SetTraceType(int type)

Parameter	Use
type	Style of the graph trace. Use either the LINES or POINTS constants

Return value:
This function has no return value.

The OOIGraphCE_SetTraceType sets the style of the graph trace. The data is displayed as either points or lines connecting the points.

OOIGraphCE_SetXData

Header: OOIGraphCE.h
void OOIGraphCE_SetXData(float* data)

Parameter	Use
data	A pointer to the data array

Return value:
This function has no return value.

The OOIGraphCE_SetXData function sets the data array that contains the X-axis values for the dataset graphed. Call this function only once, as long as you are graphing data in the same array each time the graph is redrawn. The data array pointed to by the data variable must be PIXELS (defaulted to 2048) long, or an access violation occurs. You can change the number of points in the dataset by calling OOIGraphCE_SetNumValues.

OOIGraphCE_SetXRange

Header: OOIGraphCE.h
void OOIGraphCE_SetXRange(float min, float max)

Parameter	Use
min	The minimum value for the X-axis
max	The maximum value for the X-axis

Return value:
This function has no return value.

The OOIGraphCE_SetXRange function specifies the minimum and maximum value for the X-axis. You must ensure that the minimum value is less than the maximum value.

OOIGraphCE_SetXTicks

Header: OOIGraphCE.h
void OOIGraphCE_SetXTicks(int ticks)

Parameter	Use
ticks	The number of tick marks drawn across the X-axis

Return value:
This function has no return value.

The OOIGraphCE_SetXTicks function sets the number of tick marks placed along the X-axis. For example, a value of 3 places one tick mark on the left extreme of the axis, one on the right extreme of the axis, and one in the middle of the axis.

OOIGraphCE_SetYData

Header: OOIGraphCE.h
void OOIGraphCE_SetYData(float* data)

Parameter	Use
data	A pointer to the data array

Return value:
This function has no return value.

The OOIGraphCE_SetYData function sets the data array that contains the Y-axis values for the dataset graphed. Call this function only once, as long as you are graphing data in the same array each time the graph is redrawn. The data array pointed to by the data variable must be PIXELS (defaulted to 2048) long, or an access violation occurs. You can change the number of points in the dataset by calling OOIGraphCE_SetNumValues.

OOIGraphCE_SetYRange

Header: OOIGraphCE.h
void OOIGraphCE_SetYRange(float min, float max)

Parameter	Use
min	The minimum value for the Y-axis
max	The maximum value for the Y-axis

Return value:
This function has no return value.

The OOIGraphCE_SetYRange function specifies the minimum and maximum value for the Y-axis. You must ensure that the minimum value is less than the maximum value.

OOIGraphCE_SetYTicks

Header: OOIGraphCE.h
void OOIGraphCE_SetYTicks(int ticks)

Parameter	Use
ticks	The number of tick marks drawn across the Y-axis

Return value:
This function has no return value.

The OOIGraphCE_SetYTicks function sets the number of tick marks placed along the Y-axis. For example, a value of 3 places one tick mark on the bottom extreme of the axis, one on the top extreme of the axis, and one in the middle of the axis.

OOIGraphCE_ShowCursor

Header: OOIGraphCE.h
void OOIGraphCE_ShowCursor(int show)

Parameter	Use
show	Indicates if the cursor is displayed upon the next graph redraw. Use the XONLY, YONLY, BOTH or NEITHER constants

Return value:
This function has no return value.

The OOIGraphCE_ShowCursor function toggles the presence of the cursor in the graph. The state of the cursor is not altered until the next call to OOIGraphCE_Redraw. The cursor can be vertical only (XONLY), horizontal only (YONLY), both horizontal and vertical (BOTH) or hidden (NEITHER).

OOIGraphCE_ShowGrid

Header: OOIGraphCE.h
void OOIGraphCE_ShowGrid(BOOL show)

Parameter	Use
show	TRUE to draw the grid, FALSE to not draw the grid

Return value:
This function has no return value.

The OOIGraphCE_ShowGrid function allows for the display of both horizontal and vertical grid marks on the graph. Set the spacing of the grids with the functions OOIGraphCE_SetXTicks and OOIGraphCE_SetYTicks.

Description of Miscellaneous Graphics Functions

atodouble

Header: OOIMiscFunctionsCE.h
double atodouble(LPCSTR string)

Parameter	Use
string	A pointer to a string

Return value:
The value of the double-precision floating-point number, except when the representation would cause an overflow, in which case the function returns +/–HUGE_VAL. The sign of HUGE_VAL matches the sign of the value that is not represented. The function can also return a 0 if no conversion can be performed or if an underflow occurs.

The atodouble function converts a null-terminated string into a double-precision floating point value.

atofloat

Header: OOIMiscFunctionsCE.h
float atofloat(LPCSTR string)

Parameter	Use
string	A pointer to a string

Return value:
The value of the single-precision floating-point number, except when the representation would cause an overflow, in which case the function returns +/–HUGE_VAL. The sign of HUGE_VAL matches the sign of the value that is not represented. The function can also return a 0 if no conversion can be performed or if an underflow occurs.

The atofloat function converts a null-terminated string into a single-precision floating point value.

watodouble

Header: OOIMiscFunctionsCE.h
double watodouble(WCHAR* string)

Parameter	Use
string	A pointer to a wide (double-byte or UNICODE) string

Return value:
The value of the double-precision floating-point number, except when the representation would cause an overflow, in which case the function returns +/–HUGE_VAL. The sign of HUGE_VAL matches the sign of the value that is not represented. The function can also return a 0 if no conversion can be performed or if an underflow occurs.

The watodouble function converts a null-terminated wide or UNICODE string into a double-precision floating point value.

watofloat

Header: OOIMiscFunctionsCE.h
float watofloat(WCHAR* string)

Parameter	Use
string	A pointer to wide (double-byte or UNICODE) a string

Return value:
The value of the single-precision floating-point number, except when the representation would cause an overflow, in which case the function returns +/–HUGE_VAL. The sign of HUGE_VAL matches the sign of the value that is not represented. The function can also return a 0 if no conversion can be performed or if an underflow occurs.

The watofloat function converts a null-terminated wide or UNICODE string into a single-precision floating point value.

ColorToIndex

Header: OOIMiscFunctionsCE.h
int ColorToIndex(COLORREF color)

Parameter	Use
color	An RGB color whose index is sought

Return value:
The index of the specified RGB color. If the RGB color is undefined, and index of 19 (the color white) is returned.

The ColorToIndex function converts an RGB color into an index. See the StandardColors.h header file for defined colors and indices.

GetFreeDataMemory

Header: OOIMiscFunctionsCE.h
DWORD GetFreeDataMemory(void)

Parameters:
This function requires no parameters.

Return value:
The amount of available storage or data memory available, in bytes.

The GetFreeDataMemory function returns the amount of available memory allocated to data storage.

GetFreeProgramMemory

Header: OOIMiscFunctionsCE.h
DWORD GetFreeProgramMemory(void)

Parameters:
This function requires no parameters.

Return value:
Yhe amount of available program memory available, in bytes.

The GetFreeProgramMemory returns the amount of available memory allocated to applications.

IndexToColor

Header: OOIMiscFunctionsCE.h
COLORREF IndexToColor(int index)

Parameter	Use
index	A color index

Return value:
The RGB color corresponding to the specified color index.

The IndexToColor function converts a color index into an RGB value. See the StandardColors.h header file for defined colors and indices.

Description of Communications Functions

OOIComCE_ClearRXBuffer

Header: OOIComCE.h
short OOIComCE_ClearRXBuffer(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_ClearRXBuffer function immediately clears all characters in the receive buffer. Data that was in the receive buffer before this function was called is irretrievably lost.

OOIComCE_ClearTXBuffer

Header: OOIComCE.h
short OOIComCE_ClearTXBuffer(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_ClearTXBuffer function immediately clears all characters in the transmit buffer. Data that was in the receive buffer before this function was called is irretrievably lost.

OOIComCE_Close

Header: OOIComCE.h
COMPORT OOIComCE_Close(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_Close function closes the specified communication port. Internal buffers are freed, and the communication port is left in an uninitialized state.

OOIComCE_GetBytesInRXBuffer

Header: OOIComCE.h
DWORD OOIComCE_GetBytesInRXBuffer(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_GetBytesInRXBuffer returns the number of bytes that are in the receive buffer.

OOIComCE_GetBytesInTXBuffer

Header: OOIComCE.h
DWORD OOIComCE_GetBytesInTXBuffer(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_GetBytesInTXBuffer returns the number of bytes that are in the transmit buffer waiting to be sent.

OOIComCE_GetCommError

Header: OOIComCE.h
DWORD OOIComCE_GetCommError(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if there is no pending error, otherwise on of the ERROR_ constants.

If any OOIComCE_ function returns a value other than ERROR_NO_ERROR, call the OOIComCE_GetCommError function to retrieve additional information about the error.

OOIComCE_GetCommEventMask

Header: OOIComCE.h
short OOIComCE_GetCommEventMask(COMPORT port, COMEVENTMASK* mask)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
mask	A pointer to a COMEVENTMASK variable to receive the mask

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_GetCommEventMask function retrieves the current events being monitored for the specified communication device.

OOIComCE_GetCommState

Header: OOIComCE.h
short OOIComCE_GetCommState(COMPORT port,DCB* dcb)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
dcb	A pointer to a device control block (DCB) structure to receive the communication port state

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_GetCommState function retrieves a copy of the device control block (DCB) from Windows CE that describes the current state of the communication port. If you wish to directly alter the current communication port state, you must fill a DCB structure with a call to OOIComCE_GetCommState, directly change one or more of the members of the DCB structure, and then call the OOIComCE_SetCommState to effect the changes.

OOIComCE_IsValid

Header: OOIComCE.h
BOOL OOIComCE_IsValid(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_IsValid function tests the validity of the port parameter.

OOIComCE_NoHandshaking

Header: OOIComCE.h
short OOIComCE_NoHandshaking(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_NoHandshaking function disables all communication handshaking. All Ocean Optics spectrometers and A/D products use no handshaking, and calling this function is required before attempting communications.

OOIComCE_Open

Header: OOIComCE.h
COMPORT OOIComCE_Open(WORD portnumber, DWORD rxbuffersize, DWORD txbuffersize)

Parameter	Use
portnumber	The number of the serial port. Use the COM1, COM2, COM3 or COM4 constants
rxbuffersize	The desired receive buffer size, in bytes
txbuffersize	The desired transmit buffer size, in bytes

Return value:
A handle uniquely identifying the serial port. Be sure to store the return value in a variable, as it is required when calling any OOIComCE function. If the return value is INVALID_HANDLE_VALUE, the port was not opened, and no further calls to OOIComCE_ functions is permitted.

The OOIComCE_Open command opens a parallel or serial port and prepares it for data communication. Since most handheld and palm-sized PCs do not include parallel ports, this function is used most often with serial ports. The OOIComCE_Open function does not support USB communication. Calling the OOIComCE_Open function is a requisite for calling all other OOIComCE functions.

OOIComCE_ReadBuffer

Header: OOIComCE.h
DWORD OOIComCE_ReadBuffer(COMPORT port, BYTE* buffer, DWORD bufsize)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
buffer	The data transmitted
bufsize	The size of the data buffer ready to receive the data

Return value:
The number of characters successfully received. A negative return value indicates an error.

The OOIComCE_ReadBuffer reads data from the communication port's receive buffer. The data is unformatted, and presented as an array of bytes. If bufsize bytes are not available in the receive buffer, fewer than bufsize bytes are received by the application.

OOIComCE_SetBaudRate

Header: OOIComCE.h
short OOIComCE_SetBaudRate(COMPORT port, DWORD baud)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
baud	The desired baud rate

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_SetBaudRate function immediately changes the baud rate of the specified communications port. All other communication parameters are left unchanged.

OOIComCE_SetCommBreak

Header: OOIComCE.h
short OOIComCE_SetCommBreak(COMPORT port, BOOL onoff)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
onoff	The desired state of the communication port's break status

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

If the onoff parameter is TRUE, the OOIComCE_SetCommBreak function suspends character transmission for a specified communications port and places the transmission line in a break state. If the onoff parameter is FALSE, the break status is cleared and transmission resumes for the specified communications port.

OOIComCE_SetCommEventMask

Header: OOIComCE.h
short OOIComCE_SetCommEventMask(COMPORT port, COMEVENTMASK mask)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
mask	The desired communication event mask

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_SetCommEventMask function specifies a set of events monitored for a communications device. Use the OOIComCE_WaitCommEvent function to halt execution of an application until a specified event occurs.

OOIComCE_SetCommState

Header: OOIComCE.h
short OOIComCE_SetCommState(COMPORT port,DCB* dcb)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
dcb	A pointer to a device control block (DCB) structure containing the state of the communication port

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_SetCommState function sets the state of the specified communication port as described in the DCB structure. If you wish to directly alter the current communication port state, you must fill a DCB structure with a call to OOIComCE_GetCommState, directly change one or more of the members of the DCB structure, and then call the OOIComCE_SetCommState to effect the changes.

OOIComCE_SetCommunicationParameters

Header: OOIComCE.h
COMPORT OOIComCE_SetCommunicationParameters(COMPORT port, DWORD baud, BYTE bits, BYTE stop, BYTE parity)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
baud	The desired baud rate
bits	The number of data bits
stop	The number of stop bits, use the STOPBITS_ constants
parity	The desired parity, use the PARITY_ constants in the OOIComCE.h header

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_SetCommunicationParameters configure the serial port to use the specified communication parameters. When communicating with any Ocean Optics spectrometer product, the required protocol is 9600 baud (default), 8 data bits, 1 stop bit (STOPBITS_1), and no parity (PARITY_NONE). A call to this function is required after calling the OOIComCE_Open function in order to establish the proper communication protocol.

OOIComCE_SetDTR

Header: OOIComCE.h
short OOIComCE_SetDTR(COMPORT port, BOOL onoff)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
onoff	TRUE to assert DTR, FALSE otherwise

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_SetDTR function asserts or clears the data-terminal-ready line on the specified communications port. Use this function only on serial communications ports.

OOIComCE_SetRTS

Header: OOIComCE.h
short OOIComCE_SetRTS(COMPORT port, BOOL onoff)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
onoff	TRUE to assert RTS, FALSE otherwise

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_SetRTS function asserts or clears the request-to-send line on the specified communications port. Use this function only on serial communications ports.

OOIComCE_Sleep

Header: OOIComCE.h
short OOIComCE_Sleep(COMPORT port,DWORD msec)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
msec	The number of milliseconds to sleep

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_Sleep function forces the calling thread to sleep for the specified number of milliseconds.

OOIComCE_Sleep_Window

Header: OOIComCE.h
short OOIComCE_Sleep_Window(COMPORT port,DWORD msec, HWND window, BOOL yieldcontinuous)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
msec	The number of milliseconds to sleep
window	The window whose messages are processed
yieldcontinuous	TRUE if the messages for the window specified by the window parameter are continuous, FALSE if the messages are processed only once

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_Sleep_Window function provides a flexible way to suspend the execution of a thread, while forcing Windows CE to process messages for a specified window. If the yieldcontinuous parameter is TRUE, the OOIComCE_Sleep_Window function processes the messages for the specified window, forces the thread to sleep for 1 millisecond, then repeats the process until the specified number of milliseconds have passed.

OOIComCE_UseRTSCTS

Header: OOIComCE.h
short OOIComCE_UseRTSCTS(COMPORT port, BOOL onoff)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
onoff	TRUE to use RTS/CTS handshaking, FALSE otherwise

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_UseRTSCTS function enables or disables the use of RTS/CTS handshaking for the specified communications port..

OOIComCE_UseXONXOFF

Header: OOIComCE.h
short OOIComCE_UseXONXOFF(COMPORT port, BOOL onoff)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
onoff	TRUE to use XON/XOFF handshaking, FALSE otherwise

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_UseXONXOFF function enables or disables the use of XON/XOFF handshaking for the specified communications port..

OOIComCE_WaitCommEvent

Header: OOIComCE.h
short OOIComCE_WaitCommEvent(COMPORT port, COMEVENTMASK* mask)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
mask	Pointer to a variable that receives a mask indicating the type of event that occurred. If an error occurs, the value is zero

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_WaitCommEvent function suspends execution of the tread calling OOIComCE_ functions until the specified event occurs. There is no timeout associated with this thread suspension. If the event never occurs, the thread stays suspended. When the event occurs, the mask variable contains the nature of the event.

OOIComCE_WriteBuffer

Header: OOIComCE.h
DWORD OOIComCE_WriteBuffer(COMPORT port, BYTE* buffer, DWORD bufsize)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
buffer	The data transmitted
bufsize	The size of the data buffer transmitted

Return value:
The number of characters successfully transmitted. A negative return value indicates an error.

The OOIComCE_WriteBuffer function transmits data across the communication port. The data is unformatted, and is provided as an array of BYTEs.

OOIComCE_Yield

Header: OOIComCE.h
short OOIComCE_Yield(COMPORT port)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_Yield function forces the Windows CE message pump to process pending messages.

OOIComCE_Yield_Window

Header: OOIComCE.h
short OOIComCE_Yield_Window (COMPORT port,HWND window)

Parameter	Use
port	The handle to the communications port. This is the handle returned by the OOIComCE_Open function
window	The window whose messages are processed

Return value:
ERROR_NO_ERROR if successful, otherwise on of the ERROR_ constants.

The OOIComCE_Yield_Window function forces the Windows CE message pump to process pending messages for the window specified by the window parameter.

Building Applications

This section describes how to build applications using the Ocean Optics Handheld Interface Package. This section assumes that you are familiar with building projects in Microsoft Visual C++. If you need to know how to include files in the project, or how to compile an application, please consult your compiler's documentation.

During the installation process, several subdirectories were created in our OOIHIP main directory. These include:

\INCLUDE – contains header files
\LIB – contains compiled DLLs and import libraries for popular Windows CE-compatible processors
\SOURCE – contains the source code to all library functions
\SAMPLES – includes sample source code

You must make sure that the program you write can find a copy of the OOIHIP.DLL driver. The driver can reside in the project directory or in the WINDOWS directory. We recommend that you only keep one copy of the driver on your computer, as it makes upgrading the driver easier. If you are developing multiple projects, we recommend that you place the driver in the WINDOWS directory.

Building for Emulation: If the Windows CE Toolkit for Visual C++ is installed on a computer running Windows NT, you can build Windows CE projects to run in an emulation environment. It is important to note that the emulation environment does not allow for direct access of the serial port. Therefore, applications running in the emulation environment can not call any of the OOIComCE_ or SAD_ functions.

Several header files are included with the OOHIP. You need only include the header files if you are calling functions prototyped within them.

Header File	Functions
OOIComCE.h	All OOIComCE_ functions
OOIGraphCE.h	All OOIGraphCE_ functions
OOISADCE.h	All SAD_ functions
OOIRegression.h	PerformRegression function
StandardColors.h	Defines standard colors

If you wish to rebuild the OOIHIP, you must load the OOIHIP.dsw workspace into Visual C++ 6.0 or greater. This workspace defines build settings for both the handheld and palm-size PCs with all common Windows CE-compatible processors. Prebuilt DLLs and import libraries are located in subdirectories of the OOIHIP\LIB directory.

Directory	Contents
PPC_MIPS	MIPS-based palm-size PC
PPC_SH3	SH3-based palm-size PC
HPC_ARM	ARM-based handheld PC
HPC_MIPS	MIPS-based handheld PC
HPC_SH3	SH3-based handheld PC
HPC_SH4	SH4-based handheld PC

If you wish to add support for a processor not listed here, simply add all of the C source files in the OOIHIP\SOURCE directory into a Visual C++ project and compile the project into a DLL.

Using SAD Functions

The SAD_ functions are designed to allow easy access to the advanced features of the Ocean Optics SAD500, USB2000 and Palm-SPEC. To access the ability to set and retrieve acquisition parameters, initiate spectral acquisitions and read data from memory, you must use the SAD Acquisition Mode. The functions that operate in this acquisition all begin with SAD_.

Several steps are required to utilize this acquisition mode.

1. Call SAD_Init to initialize the driver, the serial port and the hardware.
2. Configure data acquisition using SAD_ functions (e.g., SAD_SetIntegrationTime, SAD_SetSpectrometerChannel, etc.).
3. Acquire spectral data with SAD_Scan or SAD_ScanWithAverage.
4. Call SAD_Close to close the serial port when you have acquired all of your data.

Using OOIGraphCE Functions

The OOIGraphCE functions in the OOIHIP provide functions necessary to plot single-series X-Y data, as is typical with spectral datasets. In order to use the OOIGraphCE functions, simply follow these steps:

1. Call the OOIGraphCE_Init function.
2. Call the OOIGraphCE_SetNumValues function to specify the number of points in the dataset
3. Call the OOIGraphCE_SetXRange and OOIGraphCE_SetYRange to specify the extents of both the horizontal and vertical axes
4. Call the OOIGraphCE_SetXData and OOIGraphCE_SetYData functions to specify the data plotted
5. (optional) Set the tick mark spacing for the horizontal and vertical axes with OOIGraphCE_SetXTicks and OOIGraphCE_SetYTicks
6. (optional) Turn the cursor on with the OOIGraph_ShowCursor function, and sets its position with the OOIGraphCE_SetCursorPosition function
7. Call the OOIGraphCE_Redraw function to redraw the graph
8. Call the OOIGraphCE_Close to free internally allocated memory

Using OOIComCE Functions

The OOIComCE set of functions allows for easy access to serial communications via an RS-232 port. The OOIComCE library is a thin wrapper over the Windows CE serial communications functions. To use the OOIComCE library, follow these steps:

1. Call the OOIComCE_Init function to open the communication port
2. Call the OOIComCE_SetCommunicationParameters function to establish the baud rate, data bits, stop bits and parity
3. Call the OOIComCE_NoHandshaking, OOIComCE_UseXONXOFF or OOIComCE_UseRTSCTS functions to establish handshaking.
4. Call OOIComCE_ReadBuffer and OOIComCE_WriteBuffer to send and receive data.
5. Call OOIComCE_Close to terminate the serial communication session

Appendix A: SAD500 Debug Commands

SAD_ChangeSlowReadPointer

Header: OOISADCE.h
short SAD_ChangeSlowReadPointer(WORD ptr)

Parameter	Use		
ptr	the new pointer position	0 = start of memory	65535 = end of memory

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an “E” to the SAD500 followed by the new pointer position.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

SAD_ClearSlowMemory

Header: OOISADCE.h
short SAD_ClearSlowMemory(WORD block,WORD page)

Parameter	Use
block	Slow memory block
page	Slow memory page

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “c” to the SAD500 followed by the block and page of memory cleared.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

SAD_EnterDoubleSecretMode

Header: OOISADCE.h
short SAD_EnterDoubleSecretMode(WORD key)

Parameter	Use
key	The security key

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends commands to the SAD500 that enables special debugging modes.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

SAD_GetDebugMode

Header: OOISADCE.h
WORD SAD_GetDebugMode(void)

Return Value:
This function returns 1 if the SAD500 is in debug mode, 0 otherwise.

SAD500 Communication Sequence:
This function sends a "?d" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is 1 if it is in debug mode.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

SAD_GetNumberOfBadMemoryBlocks

Header: OOISADCE.h
WORD SAD_GetNumberOfBadMemoryBlocks(void)

Return Value:
This function returns the number of bad or corrupt slow memory blocks in the SAD500.

SAD500 Communication Sequence:
This function sends a "?n" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is the number of bad slow memory blocks.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

SAD_GetPrintDebugMode

Header: OOISADCE.h
WORD SAD_GetPrintDebugMode(void)

Return Value:
This function returns 1 if the SAD500 is in print debug mode.

SAD500 Communication Sequence:
This function sends a "?m" to the SAD500. After the SAD500 acknowledges the command, the next WORD returned from the SAD500 is 1 if it is in print debug mode.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

SAD_ReadSlowMemory

Header: OOISADCE.h
short SAD_ReadSlowMemory(WORD block,WORD page,WORD* data,WORD* status)

Parameter	Use
block	Memory block
page	Memory page
data	Pointer to 128 element WORD array
status	Pointer to 8 element WORD array

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an “r” to the SAD500 followed by the memory block and memory page read. After the SAD500 acknowledges the command, it then transmits 128 WORDs of memory, and 8 WORDs of status values.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

SAD_ToggleDebugMode

Header: OOISADCE.h
short SAD_ToggleDebugMode ()

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends a “d” to the SAD500 which toggles the debug mode.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

SAD_ToggleDebugPrintMode

Header: OOISADCE.h
short SAD_TogglePrintDebugMode ()

Return Value:
This function returns SAD_RETURN_SUCCESS upon successful completion, or one of the SAD_RETURN_ constants upon failure.

SAD500 Communication Sequence:
This function sends an “m” to the SAD500 which toggles the print debug mode.

Only use this command when you are instructed to do so by an Ocean Optics engineer.

Appendix B: SAD500 Data Compression

Transmission of spectral data over the serial port is a relatively slow process. Even at 115,200 baud, the transmission of a complete 2048 point spectrum takes around 400 msec. The SAD500 and the serial USB2000 implement a data compression routine that minimizes the amount of data transferred over the RS-232 connection. Using the "G" command (Compressed Mode) and passing it a parameter of 1 enables the data compression. Every scan transmitted by the SAD500 or serial USB2000 is then compressed. The compression algorithm is as follows:

1. The first pixel (a 16-bit unsigned integer) is always transmitted uncompressed.
2. The next byte is compared to 0x80.
   - If the byte is equal to 0x80, the next two bytes are taken as the pixel value (16-bit **unsigned** integer).
   - If the byte is not equal to 0x80, the value of this byte is taken as the difference in intensity from the previous pixel. This difference is interpreted as an 8-bit **signed** integer.
3. Repeat step 2 until all pixels have been read.

Using this data compression algorithm greatly increases the data transfer speed of the SAD500. The table below shows the data transfer speed, in milliseconds, for various light sources and baud rates. Keep in mind that these rates are for demonstration purposes only, and the speed of your computer may impact the data transfer rates.

	Comp	115 kb	% faster	57.6 kb	% faster	38.4 kb	% faster	19.2 kb	% faster	9.6 kb	% faster
dark	on	290	32.9%	426	45.2%	624	46.7%	1148	47.5%	2247	48.8%
	off	432		778		1170		2188		4391	
LS-1	on	290	32.9%	429	44.9%	624	46.6%	1141	49.6%	2192	50.1%
	off	432		779		1169		2266		4390	
HG-1	on	303	29.9%	465	40.2%	679	41.9%	1238	43.5%	2424	44.8%
	off	432		777		1169		2193		4391	

The following shows a section of a spectral line source spectrum and the results of the data compression algorithm.

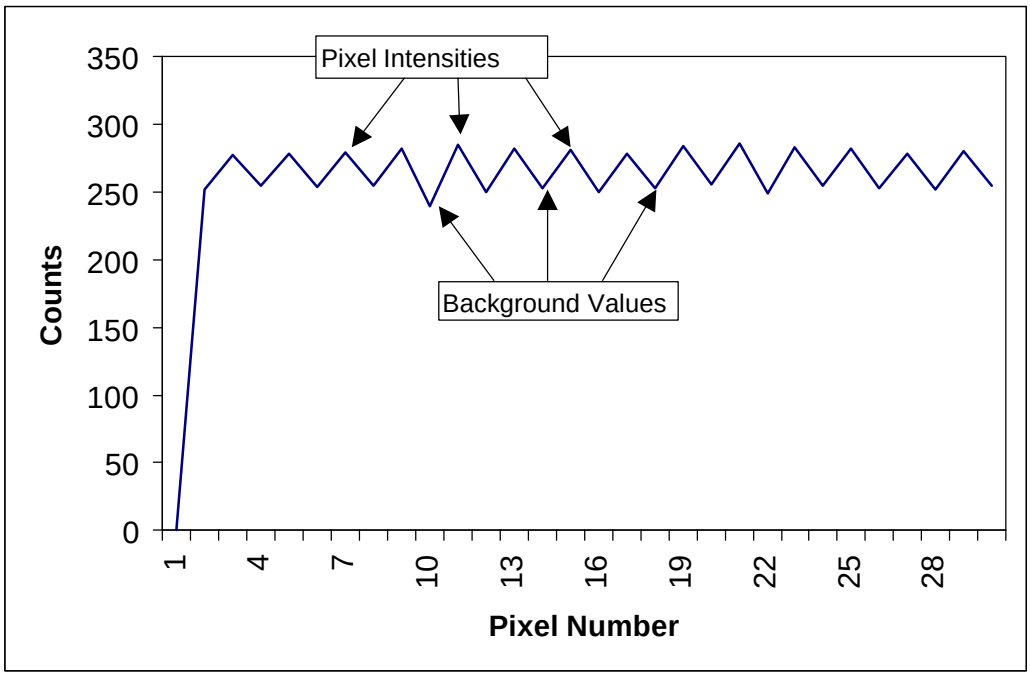
Pixel Value	Value Difference	Transmitted Bytes
185	0	0x80 0x00 0xB9
2151	1966	0x80 0x08 0x67
836	-1315	0x80 0x03 0x44
453	-383	0x80 0x01 0xC5
210	-243	0x80 0x00 0xD2
118	-92	0xA4
90	-28	0xE4
89	-1	0xFF
87	-2	0xFE
89	2	0x02
86	-3	0xFD
88	2	0x02
98	10	0x0A
121	23	0x17
383	262	0x80 0x01 0x7F
1162	779	0x80 0x04 0x8A
634	-528	0x80 0x02 0x7A
356	-278	0x80 0x01 0x64
211	-145	0x80 0x00 0xD3
132	-79	0xB1

Pixel Value	Value Difference	Transmitted Bytes
88	-44	0xD4
83	-5	0xFB
86	3	0x03
82	-4	0xFC
91	9	0x09
92	1	0x01
81	-11	0xF5
80	-1	0xFF
84	4	0x04
84	0	0x00
85	1	0x01
83	-2	0xFE
80	-3	0xFD
80	0	0x00
88	8	0x08
94	6	0x06
90	-4	0xFC
103	13	0x0D
111	8	0x08
138	27	0x1B

In this example, spectral data for 40 pixels is transmitted using only 60 bytes. If the same data set were transmitted using uncompressed data, it would required 80 bytes.

Appendix C: Correlated Double Sampling

You can configure the S1024DW and S1024DWX to utilize a data sampling technique called correlated double sampling (CDS). In this mode, each data point consists of a sampled pixel value and a sampled background value. The resulting spectral intensity is the difference between the pixel intensity and the background intensity. The figure below shows the raw output of an S1024DW.



When using the S1024DW with the SAD500, the microcontroller in the SAD500 rearranges these values so that the pixel values are transmitted first, and then the corresponding dark values.

Appendix D: SAD500 USB2000 Checksum Calculation

For all uncompressed pixel modes, the checksum is simply the unsigned 16-bit sum (ignoring overflows) of all transmitted spectral points. For example, if the following 10 pixels are transferred, the calculation of the checksum would be as follows:

Pixel Number	Data (decimal)	Data (hex)
0	15	0x000F
1	23	0x0017
2	46	0x002E
3	98	0x0062
4	231	0x00E7
5	509	0x01FD
6	1023	0x03FF
7	2432	0x0980
8	3245	0x0CAD
9	1984	0x07C0

Checksum value: 0x2586

When using a data compression mode, the checksum becomes a bit more complicated. A compressed pixel is treated as a 16-bit **unsigned** integer, with the most significant byte set to 0. Using the same data set used in Appendix B, the following shows a section of a spectral line source spectrum and the results of the data compression algorithm.

Data Value	Value Difference	Transmitted Bytes	Value added to Checksum
185	0	0x80 0x00 0xB9	0x0139
2151	1966	0x80 0x08 0x67	0x08E7
836	-1315	0x80 0x03 0x44	0x03C4
453	-383	0x80 0x01 0xC5	0x0245
210	-243	0x80 0x00 0xD2	0x0152
118	-92	0xA4	0x00A4
90	-28	0xE4	0x00E4
89	-1	0xFF	0x00FF
87	-2	0xFE	0x00FE
89	2	0x02	0x0002
86	-3	0xFD	0x00FD
88	2	0x02	0x0002
98	10	0x0A	0x000A
121	23	0x17	0x0017
383	262	0x80 0x01 0x7F	0x01FF
1162	779	0x80 0x04 0x8A	0x050A
634	-528	0x80 0x02 0x7A	0x02FA
356	-278	0x80 0x01 0x64	0x01E4
211	-145	0x80 0x00 0xD3	0x0153
132	-79	0xB1	0x00B1
88	-44	0xD4	0x00D4
83	-5	0xFB	0x00FB
86	3	0x03	0x0003
82	-4	0xFC	0x00FC
91	9	0x09	0x0009
92	1	0x01	0x0001
81	-11	0xF5	0x00F5
80	-1	0xFF	0x00FF
84	4	0x04	0x0004

Data Value	Value Difference	Transmitted Bytes	Value added to Checksum
84	0	0x00	0x0000
85	1	0x01	0x0001
83	-2	0xFE	0x00FE
80	-3	0xFD	0x00FD
80	0	0x00	0x0000
88	8	0x08	0x0008
94	6	0x06	0x0006
90	-4	0xFC	0x00FC
103	13	0x0D	0x000D
111	8	0x08	0x0008
138	27	0x1B	0x001B

The checksum value is simply the sum of all entries in the last column, and evaluates to 0x2C13.