OCT Engine Communications Command Specification

Axsun OCT Control Interface

1. Introduction

This document describes the Windows-based software interface to communicate with Axsun OCT devices including laser engines and data acquisition boards. Device control and diagnostic functions are made available to a host program via a single.dll file called "AxsunOCTControl.dll". The Microsoft .NET framework must be installed on the system.

2. Communication Protocols

The Axsun OCT control provides public methods to communicate directly with a connected device, as well as events to notify the host program if an OCT device has either connected or disconnected via USB.

Axsun OCT devices will only send data to the calling application in response to a command requesting that data.

The Axsun OCT control supports communication via Gigabit Ethernet to the Axsun Ethernet DAQ, which in turn supports pass-through communication via RS-232 interface to a Axsun laser engine board. All device control and diagnostic methods in Axsun OCT control behave in the same fashion whether the device is connected via USB, Ethernet, or RS-232 pass-through.

Note that additional methods must be called to establish an Ethernet connection and subsequently close that connection when exiting. See details below on establishing an Ethernet connection with the Axsun Ethernet DAQ.

3. Events

The Axsun OCT control uses Windows events to signal when a USB device is connected or disconnected. Use of these events to manage host program flow is optional and is not required for basic communication with an OCT device.

OCTDeviceConnectedEvent

The OCTDeviceConnectedEvent Event is called when a device is plugged into the USB or is powered on.

The event is set up to be called from a C# program like this:

```
// Create New OCT Control
AOCT = new AxsunOCTControl.AxsunOCTControl();

// Call Initialize to look for OCT USB driver and check if there is a
// device attached
AOCT.Initialize();

// Connect up the OCTDeviceConnectedEvent to a function we have locally
// called AOCT_OCTDeviceConnectedEvent
AOCT.OCTDeviceConnectedEvent += new
AxsunOCTControl.AxsunOCTControl.OCTDeviceConnectedHandler(AOCT_OCTDevic
eConnectedEvent);

// Now that the event is connected, every time an OCT device is plugged
// in, the following function will be called.
// In the OCT host program, this function queries and displays the
// firmware version in the status bar

void AOCT_OCTDeviceConnectedEvent()
{
      ...
```

OCTDeviceDisconnectedEvent

The OCTDeviceDisconnectedEvent is called whenever a USB device is disconnected from the system. Usage of this event is similar to the OCTDeviceConnectedEvent above. The line to add the event handler in C# is:

```
AOCT.OCTDeviceDisconnectedEvent += new
AxsunOCTControl.AxsunOCTControl.OCTDeviceDisconnectedHandler(AOCT_OCTDe
viceDisconnectedEvent);
```

4. Fields (Numerical Defines)

The OCT control makes public a list of numerical defines for use when programming.

USB Error Codes

The USBErrorCodes list has a list of error codes returned from the OCT control that deal with USB communication errors. There is no list of error codes for the device itself. If there is an error on the OCT device an error code will be returned that is less than 1000. The calling program can then query the OCT device for the string that defines that error code by calling GetErrorString(UInt32 ErrorCode).

For any errors that involve USB communications, however, querying the device is probably not possible and so the list of errors are provided here.

```
public const int ERROR_READING = 1000;
public const int ERROR_WRITING = 1001;
public const int NO_DEVICE_PRESENT = 1002;
public const int CHECKSUM_ERROR = 1003;
public const int TOO_MANY_PACKET_TRIES = 1004;
```

5. Methods

This is a list of methods that can be called for direct communication with OCT devices or for information about the Axsun OCT control .dll or currently connected devices.

bool CheckIfOCTDevicePresent()

Checks if an OCT device is connected. Returns true if connected, false if not.

string GetControlBuildDate()

Returns a string with the build date of the OCT control. This function does not communicate with the OCT device and the OCT device does not need to be connected for this function to return its value.

string GetControlName()

Returns the product name of the control.

string GetControlVersion()

Returns the version of just the OCT control, not the connected device(s) firmware version.

string GetErrorString()

Returns a string to describe the error matching the ErrorCode value. Error numbers and description strings are only stored onboard the OCT device. This was designed so that changes can be made to just the OCT device firmware including adding error codes without needing to change the host program or OCT control program. \

Parameters:

UInt32	Error Code	

void Initialize()

Initializes the OCT control by looking for the device driver and then, if the device driver is found and installed, checking if an OCT device is connected.

Int GetNumberOfOCTDevicesPresent ()

Returns the number of OCT devices that are connected to the system (plugged in and powered on) and enumerated in the connected device list.

Bool ConnectToOCTDevice ()

Connects to a specific enumerated OCT device in the device list. Number must be between 0 and one less than the number returned from GetNumberOfOCTDevicesPresent(). I.e. Two present devices will be enumerated as 0 and 1. If more than one device is connected and enumerated in the device list (e.g. a laser engine and a DAQ) use this function to select the connected device to which subsequent commands are directed.

Parameters:

int	DeviceNum	

Returns an error code or 0 if there was no error.

void StartPIMInterface() (Network Connected Devices only)

Establishes and maintains connectivity to a network-connected Ethernet DAQ board. Call this function once at the start of the host program operation and then wait up to 2 seconds for the DAQ device connection to be established and enumerated in the connected devices list.

void CloseConnections() (Network Connected Devices only)

Terminates connectivity to a network-connected Ethernet DAQ board. Call this function before exiting the host program.

void GetLastCommandInfo()

Returns diagnostic information about the last command sent to the host

Parameters:

ref int	BytesSent	
ref int	PacketsSent	
ref int	PacketsRetried	
ref int	Timeouts	
ref int	ChecksumErrors	
ref int	TimeTaken	
ref int	DataBytes	

UInt32 StartScan()

Start the OCT Engine laser scanning. No Parameters are passed.

Returns an error code or 0 if there was no error.

UInt32 StopScan()

Stops the OCT Engine laser scanning. No Parameters are passed.

Returns an error code or 0 if there was no error.

UInt32 RebootSystem()

Reboots connected OCT device. Useful after a firmware download. No Parameters are passed.

Returns an error code or 0 if there was no error.

UInt32 GetFirmwareVersion()

Returns a string containing the firmware version and the CPU version.

Parameters:

Ref String	Firmware Version String	

Returns an error code or 0 if there was no error.

UInt32 GetFirmwareBuildDate()

Returns the build date of the firmware.

Parameters:

Ref String	Firmware Build Date	

Returns an error code or 0 if there was no error.

UInt32 LoadFirmware()

Loads new firmware from a local array and programs it to flash.

Parameters:

Ref byte[]	Firmware image	Firmware image loaded from a .bin file. This is a .net managed array containing an array size, so no size is needed as a parameter.

Returns an error code or 0 if there was no error.

UInt32 LoadFirmwareFromFile()

Loads firmware from a file to a local array which can then be downloaded to the board with the LoadFirmware() command

Parameters:

string	Filename	Filename of .bin firmware file to open
Ref byte[]	Firmware image Array	.net managed array. This command will load this array with the firmware image.
Ref int	BytesLoaded	Number of bytes read in from the file

Returns an error code or 0 if there was no error.

UInt32 LoadFPGAFromFile()

Loads an FPGA image from a file to a local array which can then be downloaded to the ADC board with the LoadFPGA() command

Parameters:

string	Filename	Filename of .bin FPGA file to open
Ref byte[]	Firmware image Array	.net managed array. This command will load this array with the FPGA image.
Ref int	BytesLoaded	Number of bytes read in from the file

Returns an error code or 0 if there was no error.

UInt32 LoadFPGA()

Loads new FPGA image to the board and optionally programs it to flash.

Parameters:

Ref byte[]	FPGA image	FPGA image loaded from a .bin file. This is a .net managed array.
Bool	WriteToFlash	Flag to indicate whether to write this image to flash.

Returns an error code or 0 if there was no error.

UInt32 GetSystemType()

Gets the type of the currently connected board.

Type definitions are exposed in the enum OCT_SYSTEM_TYPES:

```
public const UInt32 SYSTEM_TYPE_NOT_CONNECTED = 0;
public const UInt32 SYSTEM_TYPE_OCT_LIGHT_SOURCE = 40;
public const UInt32 SYSTEM_TYPE_OCT_CAMERALINK = 41;
public const UInt32 SYSTEM_TYPE_OCT_DAQ = 42;
```

Parameters:

Ref UInt32	SystemType	Number indicating system type
Ref string	SystemTypeString	String containing the system type

Returns an error code or 0 if there was no error

UInt32 GetSerialNum()

Gets the current serial number.

Parameters:

Ref String	Serial Number	Serial Number of up to 40 characters

Returns an error code or 0 if there was no error.

UInt32 GetBootErrors()

Gets a list of any errors that occurred during system startup

Parameters:

Ref UInt32[]	Boot Errors	Array of errors. Array size must be at least 50.
Ref Uint32	Num Boot Errors	Number of boot errors that occurred

Returns an error code or 0 if there was no error.

UInt32 GetSystemStatus()

Gets an array containing system Status.

Parameters:

Ref UInt32[]	Status Arrayof size 8	Uptime - ms system has been powered on Sweeping - 1 = sweeping, 0 = not sweeping DMAError - a DMA error has occurred PPIError - a PPI error has occurred TECError - a TEC error has occurred PPIErrors - number of PPI errors since power on Boot Reg 1 – Reason for last Reboot Boot Reg 2– Reason for last Reboot
Ref Uint32	Status Words	Size of status array

Returns an error code or 0 if there was no error.

UInt32 GetUSBInfo()

Gets an array containing USB Information.

Parameters:

Ref UInt32[]	Status Arrayof size 5	PacketsRead - number of packets since power on ChecksumErrors - number of checksum errors since power on PacketErrors - number of packet errors since power on TimeoutErrors- number of timeout errors since power on CommandsReceived - number of commands recieved since power on ConnectCount - number of connections made since power on
Ref Uint32	Words Returned	Size of USB status array

Returns an error code or 0 if there was no error.

Bool IsOCTDeviceConnected ()

Checks if a device is connected.

Returns true if connected, false if not.

UInt32 GetLowSpeedADData()

Returns the 16 channels of low speed A→D data. The data is returned as 3 arrays of data.

Parameters:

Ref UInt32[]	RawVals	Raw integer data in 16 bit integers
Ref Single[]	ScaledVals	Converted values in 32 bit floats
Ref String[]	FieldNames	Strings to label each channel
Ref UInt32	ChannelsReturned	Number of channels of data returned

Using this technique, the firmware can add or delete A→D channels without needing new host software.

Currently, the channels returned are:

0	Reference Monitor
1	Signal Monitor
2	Pointer laser current
3	Filter Voltage
4	Laser Current
5	Laser TEC thermistor
6	Laser TEC current
7	Laser TEC voltage
8	Board temperature
9	Ground
10	Spare
11	3.3V reference
12	-5V
13	+5V
14	15V
15	150V

Returns an error code or 0 if there was no error.

UInt32 GetTECStatus()

Gets debug information about the TEC. The most current information is returned in the data fields. History data for the last 50 seconds is also returned in the data field.

If a TEC error occurs the TEC is turned off and history data stops collecting.

Parameters:

Ref Uint32	TEC State	Current TEC State 0 TEC_UNINT 1 TEC_WARMING_UP 2 TEC_WAITING_IN_RANGE 3 TEC_READY 4 TEC_OFF_ERROR
Ref Uint32	Counts in range	Number of seconds that the TEC has been in range
Ref Uint32	Counts Warming up	Number of seconds the TEC has been warming up for.
Ref Uint32	Counts out of range	Number of seconds the TEC has been out of range after coming in range.
Ref Uint32	TEC Error	If the TEC was shut down, the reason it was shut down. 16 TEC_NEVER_GOT_TO_READY 17 TEC_WENT_OUT_OF_RANGE The string for this error code can be returned by using the GetErrorString command (109)
Ref Uint32	TEC Temp	Current temperature. This temperature is the float temperature multiplied by 100 and returned as an integer. So, a temperature of 24.174 will be returned as an integer 2417.
Ref Uint32	Points Returned	The number of data points returned in the following 3 arrays. Right now the system returns 50.
Ref Uint32[]	TEC Temps	Temperature as a 32 bit integer of the temperature multiplied by 100.
Ref Uint32[]	TEC Times	Time in milliseconds as a 32 bit integer
Ref Uint32[]	TEC States	TEC state as a 32 bit integer.

Returns an error code or 0 if there was no error.

UInt32 ResetTEC()

Resets the TEC. Turns the TEC off, sets the temperature with a D-A write, turns it on, and resets any error code and all the TEC counters. No Parameters are passed.

Returns an error code or 0 if there was no error.

UInt32 SetFPGARegister ()

Sets one FPGA register

Parameters:

UInt32	RegNum	Register number
UInt32	RegVal	Value to set register to

Returns an error code or 0 if there was no error.

UInt32 GetFPGARegisters()

Gets values for all FPGA Registers

Parameters:

Ref UInt32[]	RegVals	Array of register values
Ref string[]	FieldNames	Array of strings containing names of each register
Ref UInt32	RegistersReturned	Number of registers that were returned

Returns an error code or 0 if there was no error.

UInt32 SourceCurrentOn()

Turns on the light source current.

Parameters:
None
Returns an error code or 0 if there was no error.

UInt32 SourceCurrentOff()

Turns on the light source current.

Parameters:
None
Returns an error code or 0 if there was no error.

UInt32 GetLaserOnTime(ref UInt32 LaserOnTimeSeconds)

Returns the number of seconds that the laser has been in use for the lifetime of the laser on this system.

Parameters:

Ref UInt32	LaserOnTimeSeconds	Number of seconds that the laser has been on.

Returns an error code or 0 if there was no error.

UInt32 GetWavelengthAtSweepTrigger(ref double Wavelength)

Returns the wavelength at the sweep trigger for this system in nm.

Parameters:

Ref double	Wavelength	Wavelength in nm for this system

Returns an error code or 0 if there was no error.

6. *Error Codes*

The following table lists the error codes used in the system.

0	OK	No Error,
1	USB_CHECKSUM_ERROR	USB Checksum Error,
2	USB_TIMEOUT_ERROR	USB Timeout Error,
3	USB_READ_ERROR	USB Read Error,
4	USB_WRITE_ERROR	USB Write Error,
5	HIGHEST_USB_ERROR	Highest USB Error,
6	DAC_TABLE_TOO_LARGE	DAC Table Too Large,
7	SCAN_SETTING_COUNT_MISMATCH	Scan Setting Count Mismatch,
8	INVALID_SCAN_SETTINGS	Invalid Scan Settings,
9	INVALID_ERROR_CODE	Invalid Error Code,
10	INVALID_DA_CHANNEL	Invalid D->A Channel,
11	INVALID_DIAGNOSTIC_TYPE	Invalid Diagnostic Type,
12	INVALID_DIAGNOSTIC_IO	Invalid Diagnostic IO,
13	INVALID_CLOCK_DELAY	Invalid Clock Delay,
14	INVALID_COMMAND	Invalid Command,
15	INVALID_CHANNEL	Invalid Channel,
16	TEC_NEVER_GOT_TO_READY	TEC Never Got to Ready,
17	TEC_WENT_OUT_OF_RANGE	TEC Went Out of Range,
18	FLASH_ERR_POLL_TIMEOUT	Polling toggle bit failed,
19	FLASH_ERR_VERIFY_WRITE	Verifying write to flash failed,
20	FLASH_ERR_INVALID_SECTOR	Invalid Sector,
21	FLASH_ERR_INVALID_BLOCK	Invalid Block,
22	FLASH_UNKNOWN_COMMAND	Unknown Command,
23	FLASH_ERR_PROCESS_COMMAND	Error Processing command,

24	FLASH_NOT_READ_ERROR	Could not read memory from target,
25	FLASH_DRV_NOTAT_BREAK	The drive was not at AFP Break Ready,
26	FLASH_BUFFER_IS_NULL	Could not allocate storage for buffer,
27	FLASH_NO_ACCESS_SECTOR	Cannot access the sector,
28	FLASH_NUM_ERROR_CODES	Flash Num Error Codes,
29	FLASH_ERR_FLASH_WRITE	Flash Write Error,
30	FLASH_ERR_FLASH_VERIFY	Flash Verify Error,
31	FLASH_ERR_FILE_OVERSIZE	Flash File Oversize Error,
32	FLASH_ERR_FILE_EMPTY	Flash File Empty Error,
33	FLASH_UNKNOWN_FILE_TYPE	Flash Unknown File Type,
34	FLASH_ERR_INVALID_DAC	Flash Invalid DAC Error,
35	FLASH_ERR_FILE_CKSUM	Flash File Checksum Error,
36	SWEEP_SPEED_TOO_FAST	Sweep Speed Too Fast,
37	TRIGGER_TIMER_TOO_LARGE	Trigger too large (Must be < 1/2 sweep),

7. Example C# Code

The following code demonstrates how to use the Axsun OCT Control to connect up the **device connected** event, connect to a device via USB, request the device firmware version and get a description of any error that is returned.

```
public class Form1 : Form
{
    public AxsunOCTControl.AxsunOCTControl AOCT;
    bool OCTConnected = false;

    public Form1()
    {
        InitializeComponent();

        AOCT = new AxsunOCTControl.AxsunOCTControl();

        // The next line wires up the device connected event.
        AOCT.OCTDeviceConnectOrDisconnectEvent += new
AxsunOCTControl.AxsunOCTControl.OCTDeviceConnectOrDisconnectHandler(AOC
T_OCTDeviceConnectOrDisconnectEvent);

        this.Controls.Add(AOCT);
        ConnectToFirstDevice();

        if (OCTConnected)
        {
            Label1.Text = "OCT Connected";
            label2.Text = GetFirmwareVersion(false);
        }
    }

    // Connects to the first enumerated OCT device.
    // This may be modified to connect to another device
    // by changing the value in ConnectToOCTDevice().
    void ConnectToFirstDevice()
    {
        bool result;

        if (AOCT.GetNumberOfOCTDevicesPresent() > 0)
        {
            result = AOCT.ConnectToOCTDevice(0);
            if (result)
            {
                OCTConnected = true;
            }
            else
            {
                OCTConnected = false;
            }
        }
        else
        {
            OCTConnected = false;
        }
    }

    // Gets the firmware version from the OCT device
    public string GetFirmwareVersion()
    {
        UInt32 result;
        string FirmwareVersion = "";

        result = AOCT.GetFirmwareVersion(ref FirmwareVersion);
        ShowOCTError(result);
        return FirmwareVersion;
    }

    // Displays a message box with the string to match the
    // numerc error code
    private void ShowOCTError(UInt32 ErrorVal)
    {
        string ErrorString;

        if (ErrorVal != 0)
        {
            ErrorString = AOCT.GetErrorString(ErrorVal);
            MessageBox.Show(ErrorString + " (" + ErrorVal.ToString() +
")", Application.ProductName);
        }
    }

    // This event is called whenever a device is either
    // plugged in or unplugged.
    void AOCT_OCTDeviceConnectOrDisconnectEvent()
    {
        ConnectToFirstDevice();
    }
}
```