In collaboration with

AOK1, AOK2, AOK3, AOK4
Adaptive Optics Kit

User Guide
Table of Contents

Chapter 1  Warning Symbol Definitions ................................................................. 5
Chapter 2  Safety ..................................................................................................... 6
Chapter 3  Description ............................................................................................ 8
  3.1. System Description ....................................................................................... 8
  3.2. Shack-Hartmann Wavefront Sensor .............................................................. 9
  3.3. Deformable Mirror ..................................................................................... 10
  3.4. Control Software ....................................................................................... 11
Chapter 4  System Assembly and Installation ......................................................... 12
  4.1. Deformable Mirror Mounting ...................................................................... 12
  4.2. Shack-Hartmann Assembly ........................................................................ 13
  4.3. Adaptive Optics Kit Assembly ................................................................... 15
  4.4. Software Installation .................................................................................. 22
  4.5. Shack-Hartmann Connection ...................................................................... 23
  4.6. Deformable Mirror Connection .................................................................. 24
  4.7. Setting the Line Select Switch and Fuse Replacement for the LDS5 .......... 25
    4.7.1. Setting the LDS5 AC Line Voltage ......................................................... 25
Chapter 5  AO Kit System Software ....................................................................... 26
  5.1. Configuring the Software for a Deformable Mirror .................................... 28
    5.1.1. The Menu Bar .................................................................................... 32
    5.1.2. The Tool Bar ..................................................................................... 33
  5.2. User Profile Manager .................................................................................. 34
  5.3. Deformable Mirror Window ...................................................................... 35
    5.3.1. Edit Box Array, Changing Values ......................................................... 37
    5.3.2. Working with the Graphs .................................................................... 37
    5.3.3. Pre-Defined View Orientations ............................................................ 38
  5.4. Wavefront Generator Window .................................................................... 40
    5.4.1. Zernike Polynomial Tab ...................................................................... 40
    5.4.2. Wavefront Tab .................................................................................... 41
  5.5. Shack-Hartmann Window ............................................................................ 42
    5.5.1. Spot Field Tab ..................................................................................... 43
    5.5.2. Wavefront Tab .................................................................................... 45
    5.5.3. Contour Tab ....................................................................................... 46
    5.5.4. Measured Zernike Tab ......................................................................... 47
  5.6. About window ............................................................................................. 48
# Chapter 1   Warning Symbol Definitions

Below is a list of warning symbols you may encounter in this manual or on your device.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="symbol" alt="Direct Current" /></td>
<td>Direct Current</td>
</tr>
<tr>
<td><img src="symbol" alt="Alternating Current" /></td>
<td>Alternating Current</td>
</tr>
<tr>
<td><img src="symbol" alt="Both Direct and Alternating Current" /></td>
<td>Both Direct and Alternating Current</td>
</tr>
<tr>
<td><img src="symbol" alt="Earth Ground Terminal" /></td>
<td>Earth Ground Terminal</td>
</tr>
<tr>
<td><img src="symbol" alt="Protective Conductor Terminal" /></td>
<td>Protective Conductor Terminal</td>
</tr>
<tr>
<td><img src="symbol" alt="Frame or Chassis Terminal" /></td>
<td>Frame or Chassis Terminal</td>
</tr>
<tr>
<td><img src="symbol" alt="Equipotentiality" /></td>
<td>Equipotentiality</td>
</tr>
<tr>
<td><img src="symbol" alt="On (Supply)" /></td>
<td>On (Supply)</td>
</tr>
<tr>
<td><img src="symbol" alt="Off (Supply)" /></td>
<td>Off (Supply)</td>
</tr>
<tr>
<td><img src="symbol" alt="In Position of a Bi-Stable Push Control" /></td>
<td>In Position of a Bi-Stable Push Control</td>
</tr>
<tr>
<td><img src="symbol" alt="Out Position of a Bi-Stable Push Control" /></td>
<td>Out Position of a Bi-Stable Push Control</td>
</tr>
<tr>
<td><img src="symbol" alt="Caution, Risk of Electric Shock" /></td>
<td>Caution, Risk of Electric Shock</td>
</tr>
<tr>
<td><img src="symbol" alt="Caution, Hot Surface" /></td>
<td>Caution, Hot Surface</td>
</tr>
<tr>
<td><img src="symbol" alt="Caution, Risk of Danger" /></td>
<td>Caution, Risk of Danger</td>
</tr>
<tr>
<td><img src="symbol" alt="Warning, Laser Radiation" /></td>
<td>Warning, Laser Radiation</td>
</tr>
<tr>
<td><img src="symbol" alt="Caution, Spinning Blades May Cause Harm" /></td>
<td>Caution, Spinning Blades May Cause Harm</td>
</tr>
</tbody>
</table>
Chapter 2  Safety

Precautions of a general nature should be gathered here. Wherever possible, however, safety warnings, cautions, and notes should only appear immediately before the instructions to which they apply (versus being listed in this section).

**ATTENTION USERS**

Please read the instruction manual carefully before operating the Adaptive Optics Kit System. All statements regarding safety and technical specifications will only apply when the unit is operated correctly.

This equipment is intended for laboratory use only and is not certified for medical applications, including but not limited to life support situations.

**ATTENTION**

This device can only be returned when received in the complete original packaging, including all parts and foam packing inserts. If necessary, ask for replacement packing materials.

**WARNING**

The LDS5 supplied with this system requires configuration for specific AC line voltages of 110VAC or 230 VAC nominal. DO NOT install a line cord into the power supply until the Voltage Select switch has been properly set or damage WILL occur. Refer to Part 8 of this manual or the LDS5 user’s manual for details.

**HIGH VOLTAGE WARNING!**

Before applying power to your system, make sure that the protective conductor of the 3 conductor mains power cord is correctly connected to the protective earth contact of the socket outlet. Improper grounding can cause electric shock resulting in severe injury or even death!

Shock Hazard: Voltages up to 300 V can be present on the deformable mirror (DM), packaging, electrodes, cable, or electronics driver.
ELECTROSTATIC DISCHARGE (ESD) WARNING

All types of electronic components, particularly integrated circuits, are sensitive to Electrostatic Discharge (ESD). ESD can lead to electronic equipment failure. The total operating time may be reduced if precautions against ESD damage are not taken.

ALWAYS wear an ESD wrist strap connected to an ESD terminal whenever handling the Adaptive Optics Toolkit (see Figure 1-1). One is included with the DM.

DM Damage

The DM is highly sensitive to electrostatic discharge. Always wear a Grounding Wrist Strap when handling the DM in an electro statically sensitive environment. Avoid touching the electrodes on the backside of the DM.

Multi-Driver Damage

The Driver is also highly sensitive to electrostatic discharge. Always wear a Grounding Wrist Strap when handling the Driver in an electro statically sensitive environment. Avoid touching any electrical interconnects.

![ESD wrist wrap](image)

Figure 1  ESD wrist wrap.

LASER RADIATION WARNING!

Do not, under any circumstances look into the optical output when the device is operating.

Specific System Classification for CPS180 Laser Module output: Class 3R @ 635 nm
Chapter 3 Description

Adaptive optics (AO) is a rapidly growing multidisciplinary field encompassing physics, chemistry, electronics, and computer science. AO systems are used to correct (shape) the wavefront of a beam of light. Historically, these systems have their roots in the international astronomy and US defense communities. Astronomers realized that if they could compensate for the aberrations caused by atmospheric turbulence, they would be able to generate high-resolution astronomical images. More recently, due to advancements in the sophistication and simplicity of AO components, researchers have utilized these systems to make breakthroughs in the areas of femtosecond pulse shaping, microscopy, laser communication, vision correction, and retinal imaging. Although dramatically different fields, all of these areas benefit from AO systems due to their ability to compensate for undesirable time-varying effects.

Typically, an AO system is comprised from three components: (1) a wavefront sensor, which measures optical wavefront deviations, (2) a deformable mirror, which can change shape in order to modify a distorted optical wavefront, and (3) real-time control software, which uses the information collected by the wavefront sensor to calculate the appropriate shape that the deformable mirror should assume in order to compensate for the distorted wavefront. Together, these three components operate in a closed-loop fashion, meaning any changes in the optical wavefront can be sensed and automatically compensated for by the system. In principle, this closed-loop system is fundamentally simple; it measures the phase as a function of the position of the optical wavefront under consideration, computes a correction to achieve the desired wavefront, reshapes the deformable mirror, observes the consequence of that correction, and then repeats this process over and over again as necessary if the phase aberration varies with time. Via this procedure, the AO system is able to improve optical resolution of an image by removing aberrations from the wavefront of the light being imaged.

3.1. System Description

Thorlabs has partnered with Boston Micromachines Corporation (BMC), a leading developer of advanced MEMS-based mirror products, to provide a series of adaptive optics (AO) toolkits, which are capable of real-time wavefront compensation by the implementation of a close loop system incorporating a Shack-Hartmann Wavefront Sensor, a MEMS deformable mirror, and control software specially created for this purpose. The AO system includes a WFS150-5C or WFS10-5C Shack-Hartmann Wavefront Sensor, which measures wavefront aberrations, BMC's 140 actuator Multi or 32 actuator Mini MEMS deformable mirror system, which can change shape in order to modify an optical wavefront, and real-time control software, which uses the information collected by the wavefront sensor to calculate the appropriate shape the deformable mirror should assume in order to compensate for aberrations. Together, these three components operate in a closed-loop configuration with a frame rate dependent upon the wavefront sensor selected to minimize aberrations, thereby increasing image resolution.

Figure 2 AO kit Schematic showing the major components included with the kit.

L, M, DM, BS, and BD refer to lens, mirror, deformable mirror, beam splitter, and beam dump, respectively. The X denotes the location of an image plane where the user can insert a sample.
3.2. Shack-Hartmann Wavefront Sensor

The WFS150-5C Shack-Hartmann wavefront sensor is equipped with a chrome mask microlens array (Thorlabs part number MLA150-5C), which is AR coated for the 300 to 1100 nm range, that enables accurate, high-speed wavefront measurements of the beam shape and intensity distribution. This is done by analyzing the location and intensity of the spots (spot field) formed by imaging the far-field wavefront of a beam of light onto a CCD with via a microlens array mounted in front of it. With Thorlabs’ Shack-Hartmann wavefront sensor, it is possible to measure the wavefronts of laser sources, characterize the wavefront aberrations caused by optical components, and provide real-time feedback for the control of the deformable mirror. The Shack-Hartmann Wavefront Sensor consists of a high-resolution (1.3 Megapixels) CCD camera with a USB 2.0, a microlens array as previously described, and analysis software. The SM1-threaded interface on the front of the CCD camera allows for the convenient mounting of ND filters, which help prevent camera saturation, and lens tubes, which reduce scattered light and allow for the mounting of additional optical components. The SM1 thread interface may be removed if one wishes to interface C-Mount style optical elements.
Thorlabs’ Fast Shack-Hartmann Wavefront Sensors, which incorporate CMOS cameras capable of providing frame rates up to 450 Hz, provide accurate measurements of the wavefront shape and intensity distribution of beams. The wavefront sensors provided with the kit come with either a chrome-masked microlens array for use in the 300 to 1100 nm range and is designed especially for higher spot field contrast.

![Thorlabs Fast Shack-Hartmann Wavefront Sensor WFS10-5C](image1)

**Figure 5** Fast Shack-Hartmann Wavefront Sensor WFS10-5C

### 3.3. Deformable Mirror

Micro-electro-mechanical (MEMS) deformable mirrors are currently the most widely used technology in wavefront shaping applications given their versatility, maturity of technology, and the high resolution wavefront correction they afford. The popular and versatile Multi-DM or Mini-DM offers sophisticated aberration compensation in an easy-to-use package. The mirror consists of a continuous mirror membrane that is deformed by 140 or 32 electrostatic actuators (i.e., a 12 x 12 or 6 x 6 actuator array with four inactive corner actuators), each of which can be individually controlled. These actuators provide 3.5 µm of stroke over a compact area; i.e., there is low influence on neighboring actuators. Unlike piezoelectric mirrors, the electrostatic actuation used with BMC’s mirrors ensures deformation without hysteresis. Each deformable mirror incorporates a window with a 6° wedge to protect the mirror. The window has a broadband AR coating for the 400 to 1100 nm range. There are eight AO Kits available, one for each DM type. For more information on specific kits, see Chapter 9 on page 61.

![BMC’s MEMS Deformable Mirror Module](image2)

**Figure 6** BMC’s MEMS Deformable Mirror Module

DM140-35-Ux01 Multi or DM32-35-Ux01 Mini
3.4. **Control Software**

Note, the Control Software is explained in detail in Chapter 5 on page 26 of this manual.

The control software provided with the adaptive optics kit is capable of minimizing wavefront aberrations by analyzing the signals from the Shack-Hartmann wavefront sensor and using those signals to determine the appropriate drive signals to send to the deformable mirror actuators so that the mirror can compensate for wavefront aberrations. The control software allows the user to monitor the wavefront corrections and intensity distribution in real time. In addition, user-defined aberrations can be introduced via the software, and wavefront deviations can be compared to this new user-defined reference.

![Software Display](image-url)

*Figure 7  Software Display*
Chapter 4  System Assembly and Installation

4.1.  Deformable Mirror Mounting

ATTENTION

An ESD wrist strap should be worn at all times while handling the DM or drive electronics. Always handle the mirror in a statically sensitive manner by using a grounded wrist strap and avoiding contact with the electrodes.

A window with an anti-reflective (AR) coating is mounted over the mirror; this is to control humidity exposure and contamination. Do NOT touch the window. AR coatings can be damaged by using improper cleaning techniques.

1. Remove the protective plastic tab covering the part on the front of the Deformable Mirror housing. The Deformable Mirror comes with a small plate attached to its bottom. Unscrew this plate with the provided Allen key HK764 to be able to attach the DM-KM1 top plate directly.

2. Screw the DM-KM1 top plate to the Deformable Mirror using the two supplied cap screws (SH6S031-1).

3. Attach the 4 TR2’s posts to the DM-KM1 mount using the supplied four setscrews (SS25S050-1). Then place them into the UPH1 post holders as shown in Figure 8.

4. Attach the Deformable Mirror with the DM-KM1 top plate to the DM-KM1 stage using the SH8S50-1 screw. Make sure the DM aperture is in line with the optical axis, which in our case is at a height of around 118 mm; this height will be obtained when all components are assembled. Make sure the TR2’s feed into the UPH1’s and touch the table surface. Fine alignment can be achieved when the setup is placed on the desired breadboard or table.

Figure 8  Deformable Mirror Assembly
4.2. Shack-Hartmann Assembly

The Shack-Hartmann sensor should be mounted following the following instructions (see Figure 9 below).

1. Secure the KM100SH to the TR3 post using the SH8S025-1 screw, and then place the post into the UPH2 post holder.

2. Mount the WFS150-5C on the T-shaped piece (which is part of the KM100SH) using the 4 screws included with the mount.

3. Attach the WFS150-5C mounted on the T-shaped base plate to the KM100SH using the 2 screws provided.

![Shack-Hartmann Wavefront Sensor Mount Assembly](image)

*Figure 9 Shack-Hartmann Wavefront Sensor Mount Assembly*
Next, follow the instructions to assemble the beam splitter and the alignment plate (see Figure 10 below):

1. Screw the BP108 pellicle beam splitter into the KM100BP mount using the two supplied screws. **NOTE: The pellicle beam splitters are extremely fragile; do NOT touch the thin surface membrane or the beam splitter will be destroyed!**

2. Secure the mount to the TR3 post with an SH8S025-1 screw, and insert the post into a UPH2 post holder.

3. Screw the SM1A7 alignment plate into the LMR1. The retaining ring inside the LMR1 may be used to secure the alignment plate with the crosshairs located at the desired orientation. Attach the LMR1 to the TR2 post, and insert the post into a UPH2 post holder.

![Figure 10  Beam Splitter and Alignment Plate Assembly Procedure](image-url)
4.3. Adaptive Optics Kit Assembly

The AO Kit has five distinct subassemblies (see Figure 12, page 16): the deformable mirror mounted (subassembly 3), three preassembled cage sections that contain the necessary optical elements (Figure 11 below and Figure 12 parts 1, 2, and 4 on page 16). The optical elements are pre-aligned such that the planar wavefront at the output of the laser will be imaged onto the sample plane (i.e., the location of the CB1), the DM surface, and the lenslet array of the Shack-Hartman sensor. Mounting posts and bases are included with the AO Kit to allow the system to be assembled on a breadboard or an optical table. Below is a description of the constituent components for each of the four subassemblies that form the AO kit system. Figure 12 shows how these four subassemblies are integrated to form the completed system.

Subassemblies 2 and 4 are for AOK1 series. Subassemblies 2 and 4 are for AOK2 Series and are Longer than Shown

Figure 11 The Three Cage System Subassemblies (Parts 1, 2, and 4).
**Figure 12 AO Kit Components**

<table>
<thead>
<tr>
<th>Part #</th>
<th>QTY</th>
<th>Item #</th>
<th>Part #</th>
<th>QTY</th>
<th>Item #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>CXY1</td>
<td>17</td>
<td>1</td>
<td>KM100BP</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>ER6</td>
<td>18</td>
<td>1</td>
<td>BP108</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>Multi: LA1608-A (or -B) Min: LA1608-A (or -B) and LA1433-A (or -B)</td>
<td>19</td>
<td>4</td>
<td>RB2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>KCB1</td>
<td>20</td>
<td>4</td>
<td>RS2</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>ER3</td>
<td>21</td>
<td>1</td>
<td>KM100SH</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>CP02</td>
<td>22</td>
<td>1</td>
<td>AKAP-1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>SM1RR</td>
<td>23</td>
<td>1</td>
<td>DM-KM1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>CB1</td>
<td>24</td>
<td>4</td>
<td>SH8S025-1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>AD11F</td>
<td>25</td>
<td>1</td>
<td>LMR1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>CPS180</td>
<td>26</td>
<td>1</td>
<td>SM1A7</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>Multi: ER1.5</td>
<td>27</td>
<td>1</td>
<td>SM1A9</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>Mini: ER4</td>
<td>28</td>
<td>1</td>
<td>NE20A</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>TR3</td>
<td>29</td>
<td>2</td>
<td>CPA1</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>CP02B</td>
<td>30</td>
<td>1</td>
<td>NE10A</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>UPH2</td>
<td>31</td>
<td>4</td>
<td>ER1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>WFS150-5C or WFS10-5C</td>
<td>32</td>
<td>4</td>
<td>ERSCA</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>BB1-E02 or BB1-E03</td>
<td>33</td>
<td>1</td>
<td>DMXXX-35-UX01</td>
</tr>
</tbody>
</table>

**NOTE:** For AOK2 series some of the opto-mechanics might look slightly different from Figure 12 (in particular sub-assembly part 4)
First, add all the TR3 Posts and UPH2 Post Holders to the three sub-assemblies as shown in Figure 12, on page 16. Sub-assembly 1 needs three TR3 Posts and UPH2 Post Holders, while sub-assemblies 2 and 4 need two TR3 Posts and UPH2 Post Holders.

Connect the first cage subassembly, part 1, with the second subassembly, part 2, as shown in Figure 13 below. Secure the ER-rods of the second subassembly using the setscrews on the CB1 plate such that the ends of the rods are flush with the inside surface of the CB1 sidewall.

After the two parts are connected make sure the optical axis is at 118 mm from the optical table or breadboard where you are building the set up. This will make the assembly match the height of the deformable mirror aperture.

Figure 13  Connecting the First Two Preassembled Cage Subassemblies (Parts 1 and 2).
The CXY1 lens mount of the second subassembly, part 2 from Figure 13, should be ~141 mm from the CP02 on the first subassembly (part 1) as noted in Figure 14. Secure the CXY1 as well as the bottom two ER6 rods (i.e., the ones in the CP02B cage plate adapters). Place the CPA1 alignment plate provided into the cage assembly between the two CP02B cage plate adapters and turn on the source. If the beam is not centered, use the XY adjustments on the CXY1 located after the CB1 to center the beam. The beam exiting the system was pre-aligned so that it should be collimated. The position of the rightmost CP02 can be adjusted to re-achieve collimation if necessary.

![Diagram](image)

**Figure 14  Achieving the Correct Optic Spacing.**

The cage-compatible angular alignment plate, shown in Figure 15, may be used to properly position the third cage subassembly (part 4) relative to the subsystem shown in Figure 14 (i.e., cage assemblies 1 and 2). Make sure the sub-assembly 4 is already fixed at the same height of the sub-assemblies 1 and 2 (~118 mm from the optical axis to the optical table). Place the angular alignment plate onto the cage systems and align the subassembly until the front surfaces of the alignment plate come in contact simultaneously with the surfaces of the CXY1 (on the third cage assembly) and the CP02 (on the second cage assembly) and the ER-rods of the subassemblies sit properly in the V-grooves in the bottom side of the alignment plate. This alignment leads to a 35° angle between the cage assemblies. Once complete, secure the cage system positions by attaching the UPH2 post holders to the optical table or breadboard and then remove the plate.

**For Mini AO Kit**

Place the angular alignment plate onto the cage systems and align the subassembly until the front surfaces of the alignment plate come in contact simultaneously with the surfaces of the CP02 (on the third cage assembly) and the CP02 (on the second cage assembly) and the ER-rods of the subassemblies sit properly in the V-grooves in the bottom side of the alignment plate. This alignment leads to a 35° angle between the cage assemblies. Once complete, secure the cage system positions by attaching the UPH2 post holders to the optical table or breadboard and then remove the plate. Keep in mind that the alignment plate is for coarse alignment, you are able to change slightly the angle to achieve fine alignment.
Figure 15  Angular Alignment of the Cage Subassemblies

As shown in Figure 16, the center of the DM’s top surface should be 73 mm from the edge of the CP02 cage plate. The location of the deformable mirror surface (inside the box) from the edge of the deformable mirror box is 11.6 mm, therefore the distance from the edge of the CP02 cage plate to the edge of the deformable mirror box should be 61.4 mm. In addition, align the Deformable Mirror assembly in such a way that the reflected beam goes through the center of the cage system. To do this, place the CPA1 alignment plate into the cage subassembly in the two indicated places, and adjust the DM’s position within the RB2 mount so that the beam is aligned with the center of the alignment plate. Once it is very close to being centered, secure the DM and use the adjuster knobs on the DM kinematic mount to make fine adjustments. Once again, ensure that the beam is still centered on the alignment plate. If not, use the XY adjustments on the CXY1 to reestablish centering.
Figure 16  Aligning the DM with the Cage Systems

Place the beam splitter assembly so that the surface of the BP108 is about 35.7 mm from the edge of the CP02. Orient it about 45° relative to the incoming beam. Place the Shack-Hartmann assembly in front of the reflected beam such that there is about 37.5 mm from the beam splitter surface to the edge of the Shack Hartmann sensor (see distance as indicated on Figure 17).
Figure 17  Alignment of the Beam Splitter, Alignment Plate, and Shack-Hartmann Sensor

Alternately one can align subassembly 4, the beam splitter, and Shack Hartmann Sensor at distances more or less than 37.5 mm so long as the total distance between them is 75 mm. Finally, place the beam dump (LMR1 assembly) in front of the transmitted beam. All of the pieces should be at a height of around 118 mm from the optical axis to match the cage system and deformable mirror height.
4.4. Software Installation

ATTENTION

Please install software prior to connecting the instruments to your PC via the USB interface. This will ensure that the correct drivers will be found.

The AO Kit Software Installation requires Microsoft .NET 2.0 framework. If the software is not installed already on the host computer, the installation software will install it during setup.

1. Install the wavefront sensor (WFS) software first. For the AO Kit application to run properly the WFS software should be version 3.5 or later. The latest version of the WFS code can be downloaded from the Thorlabs website. Using the menu, select
   
   Services ➔ Downloads ➔ Beam Characterization ➔ Shack-Hartmann Wavefront Sensor

2. Install the AO Kit application provided on the CD that is delivered with the system. If the computer system does not have .NET 2.0, the first window that appears will be a notice that it is not installed and the installation can be completed by selecting the appropriate option. The redistributable files are included in the installation file.

3. Install the Boston Micromachines Corporation (BMC) deformable mirror (DM) drivers and applications. After the installation of the AOKit application ends a window will appear offering to install these drivers and applications. This is required to run the AOKit application. Note: The files will be installed under the name Cambridge Innovations, rather than Boston Micromachines. If these drivers and applications are installed already this step can be bypassed. Please note the following for installation on systems running Windows XP operating system or Windows 7 operating system.
   
   a. Windows XP and Windows 7, 32 bit – There will be one installation process. Please follow the prompts during the installation to select the features desired.

   b. Windows 7, 64 bit – There will be two separate installation processes, the 32 bit version needs to load first. The second installation will be the 64 bit version. Both versions must be installed for the software to run properly. Please follow the prompts during these installations to select the features desired.

Once the setup completes, a new desktop shortcut will be placed on your desktop, see Figure 18. The software can now be started by double clicking the AOkit2.exe icon displayed on the desktop.

![AOKit2](Image)

Figure 18 Icon Installed on Desktop
If it was installed in the default directory, the AO Kit can also be started by accessing the application from the start button.

Click Start → All Programs → Thorlabs → AOKit → AOKit2.exe.

![Thhorlabs](image)

![AOKit](image)

![AOKit2](image)

![Documentation](image)

![Example Solutions](image)

![DCU camera](image)

**Figure 19  Accessing the AOKit2.exe File from the Start Menu.**

The AOKit Software Installation routine consists of the following:

- The installation of the Visual C++ runtime files (needed to run the AOKit application)
- The AOKit software
- The electronic manual
- The AOKit API functions document
- One C++.NET example solution
- One VB.NET example solution
- Installing the Deformable Mirror drivers

4.5. Shack-Hartmann Connection

**ATTENTION**

The Wavefront Sensor will not work on a full speed USB1.1 port!

Be sure to connect the wavefront sensor to a high speed USB2.0 port. Do NOT use cables that are not explicitly suited for high-speed USB data transfer, as this can cause transmission errors and improper instrument operation!

To connect the Shack-Hartmann Wavefront Sensor, follow the steps below

1. Find the USB 2.0 high speed cable USB (A) to Mini USB (B) included with your kit.
2. Insert the mini USB connection into the mini USB port at the back of the Shack-Hartmann Sensor
3. Connect the other end of the USB cable into one of the USB slots available on your computer (Step 3).

**Important Note**

After the Shack-Hartmann is plugged in, the computer should find the new hardware and install the driver that has just been installed from the CDROM. Once the new hardware is found, there should be a green light on the back of the Shack-Hartmann to let you know that it is recognized properly by the system.
4.6. Deformable Mirror Connection

For the following steps, please refer to the figures below:

1. Connect the USB cable to the front panel of the DM driver unit.
2. Connect the other end of the USB cable (end A) into one of the USB slots available on your computer.
3. Plug in the DB-37 ribbon cable(s) from the rear of the DM Module into the appropriate receptacle on the back of the drive unit. Note that the multi-driver has four cables labeled J1 to J4 and the mini-driver module only has one ribbon cable.
4. If setting up the multi-driver module, insert the power supply cord into the Power Input at the rear panel and plug the other end into the appropriate 110/230 VAC wall outlet. The mini driver requires no external power supply.

After the Deformable Mirror is plugged in, the computer should find the new hardware and install the driver that has just been installed from the CDROM.
4.7. Setting the Line Select Switch and Fuse Replacement for the LDS5

**WARNING**

The LDS5 operates from AC voltages which are potentially harmful and could cause death. Turn the LDS5 power switch OFF and UNPLUG the AC line BEFORE attempting to perform any of the operations below.

4.7.1. Setting the LDS5 AC Line Voltage

The operating AC voltage is indicated on the red line switch located on the bottom side of the LDS5. The LDS5 systems are shipped with this set to 115 VAC. If this needs to be changed, follow the instructions below, before plugging the AC line cord into a wall outlet:

1. Using a small screwdriver, slide the voltage selector switch to display the correct voltage rating.
2. Verify that the correct voltage is shown on the switch. If not, repeat the step above.
Chapter 5  AO Kit System Software

Recognizing that software requirements for the adaptive optics toolkit would vary greatly, and to ease the
development of customer-authored applications, Thorlabs provides a fully functional stand-alone application for
immediate operation of the instrument in addition to a low-level support library easily accessible by any modern
software development environment. The remainder of this manual will detail the stand-alone software application,
while the low-level support library is documented in the API manual: “AOSystem API functions.pdf”.

The stand-alone application (AOKit2.exe) was designed to provide users with an immediate method by which to
control the adaptive optics kit in addition to providing an overview of the kit’s functional capabilities. In actuality,
this application is little more than a graphic interface into the core arithmetic and process control functions
contained within the AOSystem.dll support library mentioned above. As end users, begin incorporating the
adaptive optics kit into their own applications, the AOKit2 application will serve as a helpful reference to guide the
end user’s software development efforts.

![Software Description Diagram](image)

**Figure 21  Software Description Diagram**

After installing the Adaptive Optics Kit software, as detailed in section 4.4 on page 22, a shortcut icon
will be placed on the desktop (see icon at the right). Start the software by double-clicking this desktop
icon or by using the Windows menu:

Start → All Programs → Thorlabs → AOKit → AOKit2.exe

When the AOKit2 application is initiated, it will make contact with and initialize the AO kit instrumentation,
specifically the Shack-Hartmann wavefront sensor and the deformable mirror interface.

Should instrumentation not be found, one of the following messages will be displayed:

Ensure that a USB 2.0 cable connects the Shack-
Hartmann sensor to the PC and that the proper software
drivers have been installed.

Ensure that a USB 2.0 cable connects the DM
interface to the PC, the interface box is powered on,
and the proper software drivers have been installed.
The last operation performed during start-up is to check for and restore user interface and instrumentation settings. If this is the first time the AOKit2 application is being executed, a warning may appear indicating that such settings do not currently exist. Under this condition the software will revert to default values pre-programmed into the software. The AOKit2 software will, from then on, automatically save all current operating conditions each time the software is used so that these values can be restored the next time the software is executed.

Upon completion of the software initialization routine, the application's "parent" window will be displayed complete with a menu and toolbar as shown below. The parent window uses a multiple document interface (MDI), allowing for flexible sizing and positioning of windows resident on the parent frame, see the figure below. Each time the software is initiated all window size and position information is stored in the default values file. Figure 23 shows an example of the Deformable Mirror's interface window as well as the Wavefront Sensor's interface window arranged on the parent frame.

![Figure 22 The Parent Window of the AOKit](image)
5.1. Configuring the Software for a Deformable Mirror

To best control the deformable mirror, the software must be configured for the DM's voltage range and response coefficients.

NOTE: These values are unique to each DM; thus, they must be entered each time a new DM is introduced into the system. All values can be found on the “Deflection Curves” chart distributed with each DM (a sample chart is shown in Figure 24).

Selecting menu item Controls → DM Options will display the following window:
- **Serial Number**: The DM serial number is found at the top of the “Deflection Curves” chart, and on the back surface of the DM.
- **Deformable Mirror Type**: The software will check the present directory for any mirror files (*.mir). Depending on the DM version you have you selection will be “Control Box Mounted DMv3.mir” or “Gimbal Mounted DMv2.mir”
- **Maximum Voltage**: The maximum voltage the DM can safely operate is the right-most value along the X-axis on the deflection curve.
- **Coefficient A**: This coefficient multiplies the x2 term of the 3X3 2nd Order trend line of the deflection curve.
- **Coefficient B**: This coefficient multiplies the x term of the 3X3 2nd Order trend line of the deflection curve.
- **Coefficient C**: This coefficient is the offset term of the 3X3 2nd Order trend line of the deflection curve.
- **Expected Displacement**: Expected stroke movement of the actuators in nanometers at full voltage. This should be calculated from the fitted displacement polynomial on the chart for the 3x3 actuator curve.

Once all of the values have been entered, click the “Calculate” button to verify your results. Clicking “Calculate” will mathematically determine the expected displacement of the mirror by using the voltage and coefficient information just entered. This is an important step to follow as it will verify the validity of the newly entered values. This step is helpful in eliminating the possibility of a data entry error.

To further illustrate this important point, we have intentionally entered an approximate value of 3500 in the “Expected Displacement” field. Upon clicking “Calculate” we see the computer calculated displacement of “3511.3355” in the Actual Displacement field along with a “100.324%” correlation to that value in the % of Expected Displacement field. If a data entry error occurred, such as entering “0.244” instead of “0.0244” for Coefficient A, the error would be immediately apparent by the large disparity between the expected and actual displacement values. When all settings appear to be valid, click “OK” to accept them.
The DM Options window will automatically appear each time the AO Kit application is started. This is intended to remind the user that if the DM has been changed these parameters must be changed; thereby minimizing the chance of inadvertently damaging the DM.

**Figure 24  Typical Characterization Graph**

A similar graph to one above is delivered with each deformable mirror. From the plot, one uses the coefficients in the equation for the top plot to enter into the DM options window.

Every deformable mirror is accompanied by a chart depicting the actuator deflection curves (this is packaged with the DM). As can be seen in the graph above, actuator deflection responds nonlinearly with drive voltage. Due to this nonlinearity, the response coefficients (see red arrow in the chart above) and maximum voltage (see the red circle on the bottom right of the chart) must be defined in the software. Important note: These values are unique to each deformable mirror.

It is important to note that while the system interface operates in spatial values (deflection in nanometers), the control of the actuators is via 14-bit digital voltage signals. This digitized voltage prevents the system from generating any arbitrary deflection value. The system will resolve "in-between" values by creating a voltage that most closely matches a desired spatial deflection.
Figure 25  Low and High Deflection Response Graphs

For the 14-bit signal there are 214 possible voltage settings over a maximum operating voltage of 300V for a DM, thus on average there are 54.6 bits per 1 Volt at the DM.

As shown in Figure 25, the first graph, based on 3 x 3 actuators being controlled simultaneously, there is a change of 10.1 nm for a one volt increase, from 9 V to 10 V. The second graph shows there is a change of 20.2 nm for a one volt change, from 217 V and 218 V. So, for the same one volt increase applied to the DM there is double the movement of the actuators at the higher voltage. As such, the movement of the actuators becomes more difficult to fine tune at higher voltages.
5.1.1. The Menu Bar

The **File** submenu provides the following file operation functions:

- **User Profile Manager**: Displays a window where the available user profiles can be selected and saved. This will be covered later in this section.
- **Exit**: Exit the program.

The **Controls** submenu provides control of the OCT channel:

- **Deformable Mirror**: Displays the Deformable Mirror window (See Section 4.3).
- **Wavefront Generator**: Displays the Wavefront Generator window (See Section 4.4).
- **Calibrate System**: Opens the Calibrate System dialog window that allows the user to set the DM stroke start/stop positions and to initiate the optical calibration procedure. This characterizes the manner in which each DM actuator influences the wavefront propagating through the system (See Section 6.)
- **Load Calibration**: Loads previously saved calibration data generated during the optical calibration process.
- **Save Calibration**: After calibration the user is able to save the calculated data so that it can be restored either during application start-up or via **Load Calibration**. This prevents the need to recalibrate each time it is turned on.
- **DM Options**: Displays the Set Quadratic Coefficients window (See section 4.1)

The **Displays** submenu lists the read devices and any calculation options:

- **Shack-Hartmann**: Opens the Shack-Hartmann interface window (See Section 4.5)

The **Help** submenu lists the help related items for the user:

- **Help**: Opens this AOkit2 manual PDF file in electronic form.
- **About the AOKit2**: Provides Thorlabs contact information in addition to Deformable Mirror and the Shack-Hartmann data.
- **Show Tooltips as API calls into the AOSystem.dll**: Turns off the regular tooltips and enables API tooltips. This allows the programmers that will be using the API for this system to get an idea of what function calls have been used for a particular menu item. If there are menu items with no direct API call behind them then the tooltip was left blank and nothing pops up. The menu item maybe was used for overhead processing or the application handled the functionality and therefore there was no API function call. This is a toggled menu item and when there is a check next to it, it is turned on.
5.1.2. The Tool Bar

Now that the deformable mirror settings have been configured we can begin exploring the software controls. Of particular interest is the toolbar located just beneath the menu. This toolbar provides easy access to the most commonly used features of the software, including Start/Stop scanning, the opening of key control windows, and activating/deactivating the wavefront correction loop. Below is a short summary of each button.

1. Opens the User Profile Manager (See Section 5.2, page 34).
2. Select a user profile for screen layout and control settings. Used in conjunction with the User Profile Manager.
3. Opens the Deformable Mirror interface (See Section 5.3, page 35).
4. Opens the Wavefront Generator interface (See Section 5.4, page 40)
5. Opens the Shack-Hartmann interface (See Section 5.5, page 42). If the Shack-Hartmann device is not detected, this window will not be available.
6. **Start Scan Button:** Begins operation of the main scanning loop which includes acquiring wavefront information from the WFS and, if activated, continuous control of the DM for wavefront correction.
7. **Stop Scan Button:** Halts operation of the main scanning loop.
8. Enable/Disable wavefront correction while the main scanning loop is active. The correction algorithm is described in greater detail in Section 5.4, page 40. **Note, the system MUST be calibrated in order for this to work.**
9. Governs the amplitude of the deformable mirror adjustment vector calculated for wavefront correction.
10. Use a flat wavefront as the Target reference wavefront. This is the default.
11. Use the current Zernike generated wavefront as the user target reference wavefront.
12. Use a captured wavefront as the target reference wavefront.
13. This will open the electronic copy of this User Manual.
5.2. User Profile Manager

Selecting the User Profile Manager Menu item will display the following window:

![User Profile Manager](image)

*This window allows the implementation of a saved user profile, which controls window sizing and positions.*

**Figure 26  User Profile Manager Window**

The AOKit2 was designed so that each user who wishes to operate the devices could have their own settings profile which contains all window sizes and locations as well as instrument control values, all of which is stored in *.ini configuration files.

The window depicted in Figure 26 shows the available *.ini configuration files found (usually in the same directory as the AOKit2 application). There should always be a default.ini file in this left window. If no .ini file is found when the AOKit2 is run, a new default.ini file will be created in the application’s directory.

If there is a profile you wish to include in the “Profile Selector” (covered earlier in this section), then simply click the right arrow button located between the two text boxes. This will copy the name of the selected profile into the profile selection text box (as in the example above with the User1.ini filename. To remove a selected profile, click the left arrow button located between the two text boxes. This will not delete the profile, just remove it so it doesn’t show up on the Profile Selector in the parent window’s toolbar.

The up and down arrow buttons at the right side of the window are used to move the selected profile up or down the list as it appears on the Profile Selector in the parent window’s toolbar.
Click the folder icon to browse for a new subdirectory where user profiles may be stored for this application. If there are many users, the administrator may choose to put the list of profiles into a separate subdirectory for easy maintenance. This is how to get to them for selection.

![Browse For Folder Window](image)

**Figure 27  Browse for Folder Window**

The **Save Selected Profile As...** button will read the currently highlighted profile in the right text box window and write it out to a user supplied filename. It will then be automatically added into the right text box profile selection. Use this option for creating an exact copy of a known good profile (like the default.ini file).

The **Cancel** button will cancel all changes to the profile master file.

The **OK** button will save the current user profiles (on the right side text box) so that they are available in the toolbar profile selector:

### 5.3. Deformable Mirror Window

The DM is capable of reconfiguring its surface topography to cancel the on- and off-axis monochromatic aberrations in the incident wavefront, which would otherwise result in a blurred image. The DM presently used in this kit is the Multi-DM, which is comprised of a 12 x 12 grid of electrostatic actuators on 400 µm centers, each of which can be displaced by 3.5 µm (i.e., stroke = 3.5 µm). The DM has an aluminum or gold coating and is capable of operating at approximately 4 kHz. The amount of voltage applied to each actuator determines its displacement and ultimately controls the profile of the surface of the DM membrane. Note that when any actuator displacement is changed, all actuators are set at one time; the API only has functions that work with the grid as a whole. The AOKit2 application writes this grid to the DM to change the surface of the mirror. Presently, there is no way to read back the status of the actuators from the DM.
This window is setup to have an edit box corresponding to an actuator and is used to change the surface of the mirror. This is done by directly editing the individual actuators and assigning a value to each of them. A profile of the resulting surface can be seen in the graph to the left of the edit box array.

**Figure 28  DM Actuator Value Array Window**

- This Dropdown-box contains some preset DM surface profiles that are helpful in testing a wavefront and measuring its effect on the Shack-Hartmann.

- Flatten DM Profile. Sets all of the edit boxes in the array to 0 (or very near 0.0) and updates DM actuator settings. This can be useful to toggle between a particular surface map and a flat surface to see contrast in the graph control of this window. (You must first save a surface map to contrast against).

- Loads Last Recalled or Saved profile file. This is the other button the user would use in order to toggle between a flat surface and one that was previously saved. Restore will reload the edit boxes (and actuators) to the last positions read from the selected file. If the user makes changes to these actuators, they could save the new profile or just restore/revert to the one last loaded.

- Opening a previously stored Surface Profile. The DM window has the ability to save and restore the actuator settings that have been set by the user or through the correction. To open a previously saved surface profile, click the “Open” button. After selecting the desired Surface Profile (.spf) file, the actuators will immediately snap to these positions and the surface profile will be shown in the graph.

- Saving a Surface Profile. Click on the “Save” button and enter a filename to save a surface profile.

To make changes to any of the actuators, the user would change the corresponding edit box.
5.3.1. Edit Box Array, Changing Values

To make a change to an actuator box, click into an edit box and highlight the text. Type a new number and press ENTER. The change should become immediately visible in the graph. This is not the only way to edit the value. The text box can be controlled with mouse wheel. The value can be changed by selecting the text in the box of the actuator you wish to control. Turning the wheel will increase and decrease the value. Alternatively, a single digit can be selected and adjusted. Adjusting the value passed 9 this way will increase the preceding digit.

Multiple boxes can be adjusted at the same time by either holding down the CTRL key and highlighting the desired boxes or using the SHIFT key and selecting two boxes. The rectangle formed by the two boxes will be highlighted. The desired value can then be typed directly or adjusted with the mouse wheel.

Note that the cursor must remain in the box while using the mouse wheel to change the value.

![Figure 29 Editing Non-continuous Boxes](image)

5.3.2. Working with the Graphs

The user has full control of tip/tilt, pan, zoom, and amplitude of the graphic display of the DM surface. The DM, Wavefront Generator, and Shack-Hartmann windows all contain this same graph control. To manipulate the graph, click on the particular graph and then use the following keys to perform the selected operations.

**Left Mouse Button:** Drag the chart around the graph window.
Right Mouse Button: Resizes the graph on its Z-axis which effectively amplifies minute changes.

Mouse Wheel: Zoom in or out depending on which way the mouse wheel is turned. In the example, pulling on the mouse zooms in.

Middle Button/Mouse Wheel Button: Tip/Tilt, turn, and spin the graph. This is most helpful in viewing the complete wavefront. From our original base picture, the mouse button was pressed and the whole mouse moved toward the top left corner.

5.3.3. Pre-Defined View Orientations
The icons located in the top-right of the graph window allow the user to orientate the graphic view. These include views from the Top, Bottom, Left, Right, Front, Back, Left Isometric, and Right Isometric.
Beginning with the above graph in the DM window, the 8 different views are:

- **Top**
  View the present graph from the top with front orientation on bottom edge.

- **Bottom**
  View the present graph from the bottom with front orientation on top edge.

- **Left**
  View the present graph from the left side with front orientation on right edge.

- **Right**
  View the present graph from the right side with front orientation on left edge.

- **Front**
  View the present graph from the front side.

- **Back**
  View the present graph from the backside.

- **Isometric Left**
  View the present graph as an isometric view from the left.

- **Isometric Right**
  View the present graph as an isometric view from the right.
5.4. Wavefront Generator Window

The wavefront generator provides an easy means of setting a user-defined reference wavefront. It offers two means to achieve this. The first is to allow the user the ability to define a Zernike polynomial based reference wavefront and the second is to allow the user to capture the ‘current’ wavefront at the Shack-Hartmann sensor. If the control mechanism in the parent-window toolbar is set to “Zernike” or “Captured Wavefront” (see page 33 items 11 and 12) and the correction mode is activated, the control system will attempt to adjust the deformable mirror surface such that the actual wavefront measured at the Shack-Hartmann sensor matches the shape created by the Wavefront Generator.

5.4.1. Zernike Polynomial Tab

This tab contains the controls to define a Zernike wavefront by combining user prescribed amounts of the first 36 Zernike terms. The label to the left of each numeric control is the monomial representation of the Zernike term\(^1\). Using the nomenclature in *Optical Shop Testing* by Malacara, a Zernike circular polynomials are noted as \(Z_{n,l}\), where \(n\) is the degree of, and \(l\) is the angular parameter of, the polynomial. To represent the polynomial with one parameter rather than two, Malacara defines

\[
m = \frac{(n - l)}{2},
\]

And

\[
r = \frac{n(n + 1)}{2} + (m + 1),
\]

Then \(Z_{n,l}\) would be notated as \(Z_{r}\).

Below is a screen capture of the Wavefront Generator tab, note the Zernike RMS amplitude values are \(\mu m\).

![Wavefront Generator Window](image)

**Figure 31 Wavefront Generator Window**

- **Sets the Zernike wavefront to flat.** Sets all of the edit boxes in the array to 0 (or very near 0.0) and therefore every Zernike term will be set this way too. This can be useful to be able to toggle between a particular Zernike plot and a flat surface to see contrast in the graph control of this window. (You must first save a Zernike plot to contrast against).

- **Restores the Zernike profile to the last profile loaded.** This is the other button the user would use in order to toggle between a flat surface and one that was previously loaded or saved. Restore will reload the edit boxes (and terms) to the last positions read from the selected file. If the user makes changes to these terms, they could save the new profile or just restore/revert to the last one loaded.

- **Opening a previously stored Zernike profile.** The Wavefront Generator window has the ability to save and restore the Zernike terms that have been set by the user. To open a previously saved Zernike profile, click the “Open” button. After selecting the desired Zernike Surface Profile (.zpf) file, the profile will show in the graph.

- **Saving a Zernike Profile.** To save a Zernike profile, just click on the “Save” button and enter a filename.

To make changes to any of these Zernike terms, the user would change the corresponding edit box. For detailed information on this, please refer to the section titled *Changing numbers in the Edit Box Array* earlier in this section.

The graph that is shown on the left of this Wavefront Generator window is not a static graph. It can be manipulated using the mouse. For detailed information on this, please refer to the section **5.3.2 Working with the Graphs** on page 37.

### 5.4.2. Wavefront Tab

The Wavefront tab includes controls that enable the user to capture the currently displayed wavefront in the Shack-Hartmann window and save it for use later. Once the user has a wavefront which is in the “Wavefront” tab of the Shack-Hartmann window, they can open the “Wavefront Generator” form and click on the “Wavefront” tab of this window. A duplicate of the wavefront will be displayed in the left panel and the last captured wavefront (or black, if none) will be displayed in the right panel.
Once there is a wavefront in the right panel, the user may save this to a file by pressing the Save button and entering a filename. The wavefronts can be reloaded by pressing the Open button and selecting the desired wavefront file. Once there is a wavefront present in the right pane, the user can use this as the Target Wavefront from which to compare the currently read wavefront from the Shack-Hartmann.

- **Sets the wavefront to flat.** This can be useful to be able to toggle between a particular wavefront and a flat surface to see contrast in the wavefront. (You must first save a wavefront to contrast against).
- **This is the other button the user would use to toggle between a flat surface and one that was previously saved. Restore will reload the wavefront to the last positions read from the selected file.**
- **Opening a previously stored Wavefront profile.** The Wavefront window has the ability to save and restore captured wavefronts that have been set by the user. To open a previously saved wavefront, click the “Open” button. After selecting the desired Captured Wavefronts (.cwf) file, the wavefront will be displayed in the right pane.
- **Saving a Wavefront.** To save a wavefront, just click on the “Save” button and enter a filename. The wavefront in the right pane will then be saved.

- **Capture the current wavefront.** The current wavefront appears in the left pane. The captured wavefront appears in the right pane. This is the first step in setting a user defined Target wavefront.
- **Set Captured wavefront as Target wavefront.** Use the captured wavefront (in the right panel) as the target reference wavefront to compare to the current wavefront to in the Shack-Hartmann wavefront tab.

### 5.5. Shack-Hartmann Window

The WFS150C Wavefront Sensor included in the AO Kit consists basically of a CCD camera with a microlens array mounted a defined distance in front of its sensor chip. Each microlens generates a spot onto the sensor whereas the spot centroid position depends on the wavefront gradient in front of the lens area. Each microlens of the lenslet array collects the light falling onto its aperture and generates a single spot on the detector plane (CCD or CMOS camera) that is situated one focal length behind the lenslets. The spot positions are located on the optical axis of each lens only in case the incident wavefront is planar and parallel to the plane of the lenslets. We call these the Reference Spot Positions or reference spot field.

Spot deviations $dx$ and $dy$ are determined by calculating the centroid coordinates of all detectable spots and subtracting the corresponding reference coordinates. These spot deviations are integrated within a 2-dimensional integration process that gives the wavefront $W(x,y)$.

Two wavefront sensors are available with the various AO Kits described in this manual. Each one has a approximately five different resolution settings. In this application, the WFS150 is set for 768 x 768 and the WFS10 is set for 360 x 360. At these settings, sensors have the same physical area and number of SH spots, 21 x 21.

The Shack-Hartmann window which controls and monitors the signal from the SH device consists of three tabs which are individually covered below.
5.5.1. Spot Field Tab

This tab is used to see the image of the spot field incident on the CCD of the SH. This is useful in setting the camera conditions that will affect the output, such as Gain and Exposure.

![Shack-Hartmann Window Showing Spot Field Distribution](image)

**Figure 33  Shack-Hartmann Window Showing Spot Field Distribution**

**Display Options**

- **Show Spots checkbox**: If checked, the spot field pattern detected by the CCD is displayed.
- **Spot Centroids checkbox**: If checked, the centroid (i.e., center with respect to power density) of each member of the spot field pattern is displayed.
- **Spot Intensities checkbox**: If checked, the intensity level for each spot in the spot field is indicated with a numeric value of arbitrary units. Saturation is indicated by red-colored pixels.
- **Reference Grid checkbox**: If checked, the spot field pattern corresponding to the reference wavefront is displayed if in Flat wavefront mode.
- **Reference checkbox**: If checked, the spot field pattern corresponding to the target reference wavefront is displayed if in Zernike or Captured wavefront modes.
- **Deviations checkbox**: If checked, a short line appears connecting each centroid spot with its corresponding reference spot.
CAMERA SETTINOS

- **Gain:** Analog amplification factor of the CCD sensor signal. This value can be set from 1.0 to 5.0 for the WFS150 and 1.5 to 5.0 for the WFS10 in incremental steps. It is advantageous to control brightness using the exposure time, and only adjust the gain amplification factor if the exposure control is too coarse. Otherwise, image noise will be increased needlessly.

- **Exposure:** Length of time (in milliseconds) of an image exposure. This value can be varied from 0.08 to 66.0 ms for the WFS150 and 0.33 to 500 ms for the WFS10. It is advisable to use this adjustment to control the brightness of the image, and then adjust the gain if this control is too coarse.

- **Average Count:** Number of frames used to determine the outputted data. If this setting is set higher than 1, the camera will average multiple images on a pixel-by-pixel basis and then output an image. Averaging may be helpful in low-light situations where measurement noise is a concern. Also, due to reduced intensity noise, the accuracy of detected spot centroids is increased. Setting this parameter to 3, 10, 30, or 100 leads to a normal average where the average is taken after achieving the selected number of image frames; after output, the image will not update again until the next set of 3, 10, 30, or 100 images is obtained. Alternatively, setting this parameter to 3 roll, 10 roll, 30 roll, or 100 roll leads to a rolling average. Here, the average of successive captured camera images is calculated every time a new image is retrieved. A new image is added to a buffer that already contains the averaged image and the summarized image is divided by the number of averages. Thus, output is provided after each additional frame is retrieved.

PUPIL DEFINITION

- **Active checkbox:** When checked, a magenta circle appears indicating the area being analyzed. All spots outside of the circle will not be considered during the analysis.

- **Diameter controls:** Allows the pupil diameter, in millimeters, to be changed in the x and y axis. When the small checkbox to the right of the numerical input control boxes is checked adjusting either the X or Y diameter setting will adjust the other setting keeping the pupil circular. If the user wants this elliptical then uncheck this box and adjust the diameter edit boxes separately.

- **Center controls:** Changes the center location of the pupil to a specified (x,y) position. This is a fine tune that is adjusted at 0.01 at a click.

- **Arrow Keys:** Changes the center of the pupil to a specific location, (x,y). This is the course adjustment and is adjusted at 0.1 at a click.

- **Center button:** Reverts the center of the pupil back to 0,0 position in the middle of the screen.
5.5.2. Wavefront Tab

This tab allows the user to see a graph along with information related to the wavefront.

- **Show Range checkbox**: If selected, this will display the range information on the graph in the top left corner.
- **Auto Range checkbox**: If selected, the lower and higher z-axis limits of the wavefront plot are updated automatically.
- **Cancel Tilt checkbox**: If selected, this cancels the average wavefront tilt.
- **Min Number box**: Lower z-axis limit of the wavefront plot. If Auto Range is not selected, this value will be used in the wavefront plot.
- **Max Number box**: Upper z-axis limit of the wavefront plot. If Auto Range is not selected, this value will be used in the wavefront plot.
- The **Camera Settings** and **Pupil Definition** controls are explained above on the Spot Field Tab. Note that a change to a setting within either of these will also change the corresponding setting on the Spot Field Tab.
- **Displayed Wavefront**: Choose from Actual, Desired, or Actual – Desired. The Actual wavefront is that measured by the wavefront sensor (i.e., calculated from the measured spot deviations using 2D integration). Choosing Desired will show an image of the reference wavefront. Finally, choosing Actual – Desired will show the difference between the actual and reference wavefronts. This is a measure of the error in the system's ability to match the actual wavefront with that desired.
The Contour control is explained on the following Contour Tab. Note that a change to a setting within this control will also change the corresponding setting on the Contour Tab.

5.5.3. Contour Tab

This tab allows the user to see a contour of the wavefront.

![Wavefront Contour Tab Window](image)

**Figure 34** Wavefront Contour Tab Window

- **Contour:** Choose from Actual, Desired, or Actual – Desired. These are 2D plots of the wavefront where height is indicated by color (from lowest to highest: black, magenta, violet, blue, cyan, green, yellow, orange, red, and white) and contour lines (i.e., lines of equal wavefront height) are superimposed on the image. The “Actual” contour is that measured by the wavefront sensor (i.e., calculated from the measured spot deviations using 2D integration). Choosing “Desired” will show a 2D image of the reference wavefront. Finally, choosing “Actual – Desired” will show the difference between the actual and reference contours. This is a measure of the error in the system’s ability to match the actual wavefront with that desired.

- **Color Map:** Select from preloaded color maps to display a multicolor wavefront contour.
5.5.4. Measured Zernike Tab

This tab allows the user to see the Zernike Coefficients of the present wavefront being displayed.

![Measured Zernike Tab Window](image)

**Figure 35  Measured Zernike Tab Window**

- **Fit to Zernike Order:** allows the user to set the Zernike polynomial order to fit to the measured wavefront. For orders greater than 6, the Zernike coefficients are displayed in split screen. See the table below for the number of terms per Zernike order.

The user may select from order 2 through order 10. At program start, the default setting is 4, which yields 15 terms and is highlighted in the table below. The table below lists the number of terms per Zernike order.

<table>
<thead>
<tr>
<th>Zernike Order</th>
<th>Number of Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>28</td>
</tr>
<tr>
<td>7</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>55</td>
</tr>
<tr>
<td>10</td>
<td>66</td>
</tr>
</tbody>
</table>
5.6. About window

The About window gives Thorlabs contact information and also provides the device information of the currently accessible devices found on the system. When the DM and the SH devices are found, their information will be displayed here such as model and serial numbers.

![About Window](image)

*Figure 36  About Window*
Chapter 6 Quick-Start

To get up and running quickly follow the following steps after assembling the kit and installing the software.

1. Start the AOKit2 software after installing all the components

2. This will begin by showing the DM Options window. Enter the max voltage of the DM along with the DM coefficients A, B, and C. Please refer to the DM Options window in the previous section for further details. Enter this information along with the expected displacement and then press the Calculate button. If the actual displacement is close to (or over) what was expected, then press <ENTER> to save the current information.

3. Open the Shack-Hartmann window and then go to the Toolbar and press the Start button. This will start scanning with the SH and display the results in its window.

4. Begin to align the SH until the wavefront amplitude P-V is below 1.00 μm. This can be seen on the Wavefront Tab with the Show Range checkbox checked. The amplitude will show in the upper left corner of the graph window. This could be a little tricky. Make sure that the SH is straight and that the gain and exposure settings are not turned up too high. Set both of these settings as low as possible and slowly increase the settings until reaching the desired effect/reading. For a more in depth explanation of this, please refer to the next section, The AO Kit Calibration

5. On the menu bar, go to Controls Æ Calibrate System. In the dialog window that pops up, set the start and end actuator stroke offsets, then press the Calibrate DM button in the window. Then watch as the system moves each of the DM actuators and measures its influence on the resulting wavefront. This usually takes less than a minute and then a Save As window opens up allowing the user to save the AO calibration. Enter a unique filename and press <ENTER> to save the calibration.

6. Set the DM flat by entering 0.0 in all edit boxes. To do this, press the flat button on the DM window or to do a group edit, see page 41.

7. Introduce a sample wavefront into the system and see the wavefront.

8. From the Tool Bar, press the Enable Correction button to start the correction.

9. From the Tool Bar, the user can enter a different gain value which will have an effect on how fast and accurate the wavefront will correct itself.

How the System Works

The concept is that we align the SH with the DM first to make sure there is little wavefront distortion and take a measurement of how much each DM actuator will affect the overall wavefront. The sample is introduced into the system to give a different wavefront. The correction mode is then turned on to calculate the conjugate shape needed on the DM to correct for distortion. As the DM corrects, we should see a flattened wavefront.
Chapter 7  The AO Kit Final Alignment and Calibration

After assembling the opto-mechanic system and installing the software, a final alignment procedure must be performed prior to calibrating the system for adaptive control. The following are the alignment objectives.

1. The center of the laser spot through the sample region should be transformed to the center of the deformable mirror.

2. The center of the DM should be aligned to the center of the Shack-Hartmann sensor. The active area of the DM is imaged onto the lens array of the SH sensor, such that the DM area covers approximately 32 x 32 lens area. Aligning the center of the DM with the center of the SH sensor ensures the outlying DM actuators affect SH spot positions within the 32 x 32 lens area.

Step 1. Align the center of the DM with the optic axis of the input cage system, subassemblies 1 and 2.

Figure 37  Laser Diode Module Alignment

1) Using the DM window of the AOKit application set the center four actuator values to 3000 to 3500. This will pull a small group of center actuators creating a ‘dimple’ in the center of the DM.

2) With the laser diode module powered on, place a CPA1 cage system alignment plate onto the ER rods just prior to, or just after, the CB1 U-Bench that joins subassemblies 1 and 2 (see Figure 37). The laser spot should be centered on the engraved target of the CPA1 and that there is a small visible spot incident on the DM (see Figure 37).

3) With the CPA1 set in position adjust the vertical and horizontal position of the DM such that the visible spot is centered in the dimpled region of the DM.
a) While moving the DM maintain the distance from the final lens of the input cage system.

b) When the spot is close to the desired position, it may be more convenient to adjust the TR posts’ heights and UPH1.5 post holders’ positions that support the cage system to perform the final fine adjustments.

4) Set the DM values to ‘0’ (using the Flatten button in the DM window applies ‘0’ to all actuators) and manually adjust the yaw and pitch of the DM assembly such that the beam reflecting from the DM into the final optical relay is close to propagating through then centers of the relay lenses. Although the fine alignment of the DM to the final optic relay section will be addressed in the following step, the more precisely the position of the DM is set at this point will greatly help the next step. Use a 2nd CPA1 to check the alignment of the beam propagating through the optic relay.

**Step 2.** Align the center of the DM to the optic axis of the optical relay section (subassembly 4).

---

**Figure 38  Aligning DM using the Cage System**

1) Relative to the DM, with the DM mirror flattened (DM values set to ‘0’) and a 2nd CPA1 at the proximal lens (near the DM) adjust the DM’s pitch and yaw using the adjusters on the DM stage to center the beam entering the optic relay (refer to Figure 38).

2) With the CPA1 at distal end of the relay adjust the CXY1 (x-y adjustable lens mount) at the proximal end to center the beam. If the range of adjustment of the CXY1 does not allow one to center the beam then the vertical/horizontal position of the relay should be adjusted. Center the CXY1 prior to making any vertical/horizontal adjustments to the relay.

3) Repeat steps 1 and 2 to center the beam through the final relay.
Step 3. Align the center of the DM to the center of the SH sensor.

**Figure 39  SH Sensor Alignment Diagram**

1) Ensure the beam strikes the center of the Pellicle Beam splitter. If the beam is not centered, vertical alignment can be achieved by adjusting the height of the TR post and horizontal alignment can be achieved by moving the UPH1.5 itself. If only fine alignment is required, adjust the position of the beam using the knobs on the KM100BP kinematic mount.

2) Adjust the SH sensor’s position following the same procedure for the height and fine alignment. It may be helpful to screw the SM1A7 alignment disk onto the ND filter on the front of the wavefront sensor to provide a visual marker of the center.

3) Adjust the spot distribution field of the SH such that they are centered on the spot field tab of the SH window. Figure 1 below shows how the spot distribution should be centered in the spot field tab screen.

   a) Adjust the height of the SH and position relative to the beam so that it goes through the center of the NE20A filter and onto the center of the SH’s matrix.

   b) Once the field is relatively centered on the screen, use the kinematic adjustment on the KM100BP to fine position the beam (refer to Figure 39 for knob adjustment information).
4) Monitoring the wavefront tab of the SH window (refer to Figure 1), adjust the pitch and yaw of the SH mount such that the wavefront flattens. The default setting for the wavefront display is auto-ranging so the wavefront will never appear absolutely flat; but, with all CPA1’s removed from the system the peak-to-valley of the wavefront should be less than 1 micron if things are aligned properly.

![Spot Field Not Centered](image1)

![Spot Field Centered](image2)

**Figure 1  Spot Field Distribution Window**
Step 4. Fine Shack-Hartmann sensor alignment

1) With the Shack-Hartmann Spot Field tab activated, click the boxes to the left of Reference Grid and Deviations boxes. This enables one to see how much the actual spot field is deviated relative to the reference grid, as presented in Figure 2. To align these spots, reducing the deviation to near-zero, adjust the tip and tilt knobs of the SH sensor mount until the spot fields and the reference grid are in the same position (on top of each other).

![Figure 2](image_url)  
**Figure 2**  Deviation of the Spot Fields from the Reference Grid
Step 5. Reduce amplitude PV value to the minimum.

**Figure 3 Shack-Hartmann Wavefront Windows with Amplitude Adjustments**

1) Switch from the Spot field tab to the Wavefront tab. The Gap value represents the wavefront deviation in multiples of the light’s wavelength. It has to be minimized in order to obtain accurate information of the wavefront profile. Use the Tip/Tilt adjustments on the KM100P mount holding the SH Sensor to minimize the amplitude. Figure 3 show how the amplitude value was considerably reduced via this procedure. Alternately, there is a check box titled Cancel Tip/Tilt that when check will remove the average tip and tilt of the measured wavefront.

**IMPORTANT NOTE**

The quality of the alignment can be judged by the amplitude value and the overall centricity of the beam on the SH matrix (as seen with the Spot Field Screen). A PV amplitude value from 0.2 to 1.0 signifies a good alignment. If good alignment cannot be achieved on the first try, repeat Steps 1 through 3 until good alignment is obtained.
Step 6. Pre-requisite for Deformable Mirror Calibration

1) Once the SH sensor is calibrated, toggle the Spot Intensities check box on by clicking in the appropriate box on the Spot Field tab (refer to Figure 4). Each red number on a spot field represents the relative intensity within a one microlens area. The intensities with marginal values will appear as flickering numbers, which are typically seen at the edges of the field. These flickering numbers must be eliminated in order to calibrate the DM. Change the Threshold values circled in red on Figure 4 to eliminate flickering values. Set the Threshold min above the flickering point’s intensity value so that it is not included in the wavefront calculation. The same methodology applies to the Threshold max value.
Step 7. The system is now ready for the Deformable Mirror calibration.

![Calibration System Dialog](image)

**Figure 5  Deformable Mirror Calibration System Options and Dialog Windows**

1) Select the **Calibrate System** option under the Controls menu item as shown in Figure 5.

2) This will bring up the Calibrate System Dialog which is used to set the stroke of the DM during calibration. Adjust the **Start Stroke Offset** and the **End Stroke Offset** which are the percentages of the DM actuator’s full stroke. The default settings of 50% and 100% (shown in Figure 5) would set all the actuators to 50% stroke when the user presses the **Calibrate DM** button. Then sequentially, each actuator would move to its full stroke limit determined by the values entered into the DM Options window. If the default values are changed, pressing **Apply** will set the DM to calibrate to the new values. Pressing the **Save** will save the settings shown. The **Cancel** button would revert to the settings when the dialog box was first opened. Selecting any of these buttons (other than **Apply**) will cause the dialog window to close.
3) Pressing the **Calibrate DM** button causes the dialog window to close and starts the system calibration. Each DM actuator is sequentially displaced, and for each displaced actuator the Δx and Δy displacements of each SH sensor spot is recorded. As each actuator is adjusted, a surface spike should be seen on the DM surface as shown in Figure 6. At the end of the procedure the calibration needs to be saved by selecting the **Save Calibration** option under the control menu.

![DM Calibration Window](image)

**Figure 6**  DM Calibration Window
4) A calibrated Deformable Mirror is shown in Figure 7. The system sets the effective Wavefront region (white cells) and excludes marginal points that introduce noise (red cells). These cells can be manually toggled noise/verified cell by right clicking on the desired cell.

![Calibrated DM with the Defined Effective Region](image)

**Figure 7  Calibrated DM with the Defined Effective Region**

5) Click the **Enable Correction** button, see figure below, so that the deformable mirror can constantly correct for wavefront aberrations.

![Enable Correction](image)

6) The system is now ready to be used in applications. Experimental samples are to be placed into the CB1 mount midway between the CP02 and CXY1 plates as shown in in the figure below.
Chapter 8  Care of the AO Kit System

Handle the system components with care during transportation and unpacking. Banging or dropping the components can damage or lower system performance. Important:

- Do not store or operate in a damp, closed environment.
- Do not store or operate on surfaces that are susceptible to vibrations.
- Do not expose to direct sunlight.
- Do not use solvents on or near the equipment.
- Keep away from dust, dirt, and air-born pollutants (including cigarette smoke).
- The system is not designed for outdoor use.
- Protect the equipment from rain, snow, and humidity.
- Do not expose to mechanical or thermal extremes.
- Protect the equipment from rapid variation in temperature.
- Handle all connectors, both electrical and optical, with care. Unnecessary force may damage the connectors.

Under normal operating conditions, the Shack-Hartmann wavefront sensor does not need any service. It is recommended that the dust cover be screwed on whenever the Wavefront sensor is not in use. Remove dust in the instruments entrance aperture with oil-free compressed air. There is no need to recalibrate the WFS150-5C Wavefront Sensors after a period of time.

Do not stick anything into the aperture at the front of the Wavefront Sensor! You may damage the microlens array because there is no glass covering in front of it.

8.1  Service

Only trained Thorlabs personnel should service the system. Please contact techsupport@thorlabs.com.

8.2  Accessories and Customization

Any modifications or servicing done by unqualified personnel renders the warranty null and void, leaving Thorlabs free of liability. Please contact Thorlabs for questions on customization.
**Chapter 9 Specifications**

<table>
<thead>
<tr>
<th>Item #</th>
<th>AOK1-UM01</th>
<th>AOK1-UP01</th>
<th>AOK2-UM01</th>
<th>AOK2-UP01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Boston Micromachines Deformable Mirror (Multi-DMTM)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deformable Mirror Type</td>
<td>DM140-35-Ux01 Multi-DM</td>
<td>DM32-35-Ux01 Mini-DM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator Array</td>
<td>12 x 12</td>
<td>6 x 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator Count</td>
<td>140</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator Stroke</td>
<td>3.5 µm</td>
<td>3.5 µm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator Pitch</td>
<td>400 µm</td>
<td>400 µm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear Aperture</td>
<td>4.4 mm x 4.4 mm</td>
<td>2.0 mm x 2.0 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mirror Coating</td>
<td>Gold</td>
<td>Aluminum</td>
<td>Gold</td>
<td>Aluminum</td>
</tr>
<tr>
<td>Mirror Wavelength Range</td>
<td>600 – 1100 nm</td>
<td>400 – 1100 nm</td>
<td>600 – 1100 nm</td>
<td>400 – 1100 nm</td>
</tr>
<tr>
<td>Surface Quality</td>
<td>&lt;40 nm RMS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame Rate</td>
<td>8 kHz (34 kHz Burst)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Step Size</td>
<td>&lt;1 nm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inter-Actuator Coupling, CDM</td>
<td>20 – 40%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Head Dimensions</td>
<td>4.5&quot; x 2.95&quot; x 2.8&quot; (114.3 mm x 74.9 mm x 71.1 mm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Interface</td>
<td>USB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Thorlabs' WFS150-5C Shack-Hartmann Wavefront Sensor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Type</td>
<td>CCD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wavelength Range</td>
<td>300 – 1100 nm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame Rate</td>
<td>15 Hz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camera Resolution (Max)</td>
<td>1280 x 1024 pixels (Set to 768 x 768)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pixel Size</td>
<td>4.65 µm x 4.65 µm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Lenslets (Max)</td>
<td>39 x 31 (Set at 21 x 21)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lenslet Pitch</td>
<td>150 µm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lenslet Diameter</td>
<td>146 µm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective Focal Length</td>
<td>3.7 mm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aperture Size</td>
<td>Max 5.95 mm x 4.76 mm (Set at 3.7 mm x 3.7 mm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wavelength Sensitivity</td>
<td>λ/50 rms (@ 633 nm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wavefront Dynamic Range</td>
<td>&gt;100λ (@ 633 nm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exposure Range</td>
<td>77 µs – 66 ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image Digitization</td>
<td>8 bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optical Input Connector</td>
<td>C-Mount</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical Size (H x W x D)</td>
<td>34 mm x 32 mm x 48.3 mm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Supply</td>
<td>&lt;1.5 W via USB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warm-up Time for Rated Accuracy</td>
<td>15 min</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating and Storage Temp</td>
<td>5 to 40 °C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>10 to 85%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

2 All technical data is specified at 23 ± 5 °C and 45 ± 15% relative humidity.
<table>
<thead>
<tr>
<th>Item #</th>
<th>AOK3-UM01</th>
<th>AOK3-UP01</th>
<th>AOK4-UM01</th>
<th>AOK4-UP01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Boston Micromachines Deformable Mirror (Multi-DM)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deformable Mirror Type</td>
<td>DM140-35-Ux01 Multi-DM</td>
<td>DM32-35-Ux01 Mini-DM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator Array</td>
<td>12 x 12</td>
<td>6 x 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator Count</td>
<td>140</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator Stroke</td>
<td>3.5 µm</td>
<td>3.5 µm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuator Pitch</td>
<td>400 µm</td>
<td>400 µm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear Aperture</td>
<td>4.4 mm x 4.4 mm</td>
<td>2.0 mm x 2.0 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mirror Coating</td>
<td>Gold</td>
<td>Aluminum</td>
<td>Gold</td>
<td>Aluminum</td>
</tr>
<tr>
<td>Mirror Wavelength Range</td>
<td>600 – 1100 nm</td>
<td>400 – 1100 nm</td>
<td>600 – 1100 nm</td>
<td>400 – 1100 nm</td>
</tr>
<tr>
<td>Surface Quality</td>
<td>&lt;40 nm RMS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame Rate</td>
<td>8 kHz (34 kHz Burst)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Step Size</td>
<td>&lt;1 nm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inter-Actuator Coupling, CDM</td>
<td>20 – 40%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Head Dimensions</td>
<td>4.5&quot; x 2.95&quot; x 2.8&quot; (114.3 mm x 74.9 mm x 71.1 mm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Interface</td>
<td>USB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Thorlabs' WFS10-5C Shack-Hartmann Wavefront Sensor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Type</td>
<td>CMOS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wavelength Range</td>
<td>300 – 1100 nm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame Rate</td>
<td>450 Hz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camera Resolution (Max)</td>
<td>680 x 480 pixels (Set to 360 x 360)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pixel Size</td>
<td>9.9 µm x 9.9 µm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Lenslets (Max)</td>
<td>41 x 29 (Set at 21 x 21)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lenslet Pitch</td>
<td>150 µm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lenslet Diameter</td>
<td>146 µm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective Focal Length</td>
<td>3.7 mm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aperture Size (Max)</td>
<td>6.34 mm x 4.76 mm (Set at 3.6 mm x 3.6 mm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wavelength Sensitivity</td>
<td>λ/30 rms (@ 633 nm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wavefront Dynamic Range</td>
<td>&gt;100λ (@ 633 nm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local Radius of Curvature</td>
<td>&gt;7.4 mm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exposure Range</td>
<td>33 µs – 500 ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image Digitization</td>
<td>8 bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optical Input Connector</td>
<td>C-Mount</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical Size (H x W x D)</td>
<td>37.8 mm x 44.8 mm x 54.9 mm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Supply</td>
<td>&lt;1.5 W via USB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warm-up Time for Rated Accuracy</td>
<td>15 min</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating and Storage Temp</td>
<td>5 to 40 °C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>10 to 85%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 All technical data is specified at 23 ± 5 °C and 45 ± 15% relative humidity.
## 9.1. Packing List

Please refer to the packing list below to ensure that the system is complete. Use only original parts. If any item is missing or damaged, contact Thorlabs for assistance.

<table>
<thead>
<tr>
<th>Components</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Optics Operating Manual</td>
<td>1</td>
</tr>
<tr>
<td>Shack-Hartmann Operating Manual</td>
<td>1</td>
</tr>
<tr>
<td>Preassembled Cage Section 1</td>
<td>1</td>
</tr>
<tr>
<td>Preassembled Cage Section 2</td>
<td>1</td>
</tr>
<tr>
<td>Preassembled Cage Section 4</td>
<td>1</td>
</tr>
<tr>
<td>BMC’s Deformable Mirror (mini or multi DM) and Driver Module</td>
<td>1 ea.</td>
</tr>
<tr>
<td>AKAP-1 Alignment Plate for Preassembled Cages</td>
<td>1</td>
</tr>
<tr>
<td>Components to Mount DM: TR2’s, UPH1’s, DM-KM1 and Screws</td>
<td>4 ea.</td>
</tr>
<tr>
<td>(excluding mount which is only 1)</td>
<td></td>
</tr>
<tr>
<td>Shack-Hartmann Wavefront Sensor (WFS150-5C) or Fast Shack-Hartmann (WFS10-5C) with Dust Cover</td>
<td>1</td>
</tr>
<tr>
<td>Components to Mount the Wavefront Sensor: KM100SH, TR3, UPH2, SM1A9, NE10A and NE20A</td>
<td>1 ea.</td>
</tr>
<tr>
<td>Alignment Plate (SM1A7)</td>
<td>1</td>
</tr>
<tr>
<td>Components to Mount Alignment Plate: TR3, LMR1, and UPH2</td>
<td>1 ea.</td>
</tr>
<tr>
<td>Beam Splitter (BP108)</td>
<td>1</td>
</tr>
<tr>
<td>Components to Mount Beam Splitter: KM100BP, SH85025-1, TR3 and UPH2</td>
<td>1 ea.</td>
</tr>
<tr>
<td>CD-ROM containing the Adaptive Optics Toolkit software, which includes the required driver and application software for the Deformable Mirror</td>
<td>1</td>
</tr>
<tr>
<td>CD-ROM for Shack-Hartmann wavefront sensor, which includes the required driver and application software</td>
<td>1</td>
</tr>
<tr>
<td>CPA1</td>
<td>3</td>
</tr>
<tr>
<td>USB2.0 High Speed Cable USB A to Mini USB B, 1.5 m</td>
<td>2</td>
</tr>
</tbody>
</table>
Chapter 10  Warranty Information

General Product Warranty

Thorlabs warrants that all products sold will be free from defects in material and workmanship and will conform to the published specifications under normal use, when correctly installed and maintained. However, Thorlabs does not warrant a fault free and uninterrupted operation of the unit, of the soft- or firmware for special applications nor this instruction manual to be error free. Thorlabs is not liable for consequential damages.

The customer will incur shipping costs to Thorlabs for warranty repairs; for such repairs, Thorlabs will incur the shipping costs back to the customer. If no warranty repair is applicable, the customer is responsible for shipping costs both to and from Thorlabs. In case of shipment from outside EU duties, any taxes or other fees that should arise, will be the responsibility of the customer.

Restriction of Warranty

The warranty mentioned above does not cover any damage or defects resulting from improper treatment, modifications to the unit, misuse, or operation outside of defined ambient conditions, unauthorized maintenance, or software/interfaces not supplied by Thorlabs. Further claims will not be consented to and will not be acknowledged. Thorlabs explicitly does not warrant the usability of the system for certain applications. Thorlabs reserves the right to change this instruction manual or the technical data of the described unit at any time.

Deformable Mirror

Any software or hardware failures due to manufacturing or inherent defects will be fixed by repair or replacement of defective parts (at Boston Micromachines Corporation’s option) for a period of ninety days after delivery. After ninety days, any software or hardware failures due to manufacturing or inherent defects will be fixed by repair or replacement (at Boston Micromachines Corporation’s option) for one year after delivery, at a cost to include parts and labor. Warrantee for the Deformable Mirror does not include damages incurred from mishandling or misuse of the DM or driver, including failure due to Electric Shock Discharge (ESD), excessive optical intensity, or by not following the directions in this document.

Optomechanics

Lifetime Warranty: Thorlabs offers a lifetime warranty on all optomechanical components. Thorlabs will repair or replace any optomechanical product which, after evaluation, has been shown to not meet specifications under the conditions listed.

Optical Tables and Breadboards

Lifetime Warranty: Thorlabs provides a lifetime guarantee that all of our passively damped optical tables and breadboards will meet all originally stated performance specifications under normal use and proper handling. We additionally guarantee that all our table tops and breadboards, both active and passive, will be free from defects in workmanship, including delamination of the skins under normal use and handling.

Lasers and Imaging Systems

Thorlabs offers a one year warranty on all lasers and imaging systems, with the exceptions of laser diodes. Some products are warranted for the number of hours specified in the operating manual of each laser.
Opto-Electronics, Control Electronics, Optics, and Nano-Positioning Product Lines

Thorlabs offers a two year warranty on the above mentioned product lines, provided normal use and maintenance of the products and when properly handled and correctly installed.

Thorlabs shall repair or replace any defective or nonconforming product as detailed above. We ask that buyer contact Thorlabs for a Return Material Authorization number (RMA #) from our Customer Service/Returns department in order to most efficiently process the return and/or repair.

Non-Warranty Repairs

Products returned for repair that are not covered under warranty, will incur a standard repair charge, in addition to all shipping expenses. This repair charge will be quoted to the customer before the work is performed.

Warranty Exclusions

The stated warranty does not apply to Products which are (a) specials, modifications, or customized items (including custom patch cables) meeting the specifications you provide; (b) ESD sensitive items whose static protection packaging has been opened; (c) items repaired, modified or altered by any party other than Thorlabs; (d) items used in conjunction with equipment not provided by, or acknowledged as compatible by, Thorlabs; (e) subjected to unusual physical, thermal, or electrical stress; (f) damaged due to improper installation, misuse, abuse, or storage; (g) damaged due to accident or negligence in use, storage, transportation or handling.
Chapter 11 Regulatory

As required by the WEEE (Waste Electrical and Electronic Equipment Directive) of the European Community and the corresponding national laws, Thorlabs offers all end users in the EC the possibility to return “end of life” units without incurring disposal charges.

- This offer is valid for Thorlabs electrical and electronic equipment:
  - Sold after August 13, 2005
  - Marked correspondingly with the crossed out “wheelie bin” logo (see right)
  - Sold to a company or institute within the EC
  - Currently owned by a company or institute within the EC
  - Still complete, not disassembled and not contaminated

As the WEEE directive applies to self-contained operational electrical and electronic products, this end of life take back service does not refer to other Thorlabs products, such as:

- Pure OEM products, that means assemblies to be built into a unit by the user (e.g. OEM laser driver cards)
- Components
- Mechanics and optics
- Left over parts of units disassembled by the user (PCB’s, housings etc.).

If you wish to return a Thorlabs unit for waste recovery, please contact Thorlabs or your nearest dealer for further information.

11.1. Waste Treatment is Your Own Responsibility

If you do not return an “end of life” unit to Thorlabs, you must hand it to a company specialized in waste recovery. Do not dispose of the unit in a litter bin or at a public waste disposal site.

11.2. Ecological Background

It is well known that WEEE pollutes the environment by releasing toxic products during decomposition. The aim of the European RoHS directive is to reduce the content of toxic substances in electronic products in the future.

The intent of the WEEE directive is to enforce the recycling of WEEE. A controlled recycling of end of life products will thereby avoid negative impacts on the environment.
Chapter 12  Thorlabs Worldwide Contacts

USA, Canada, and South America
Thorlabs, Inc.
56 Sparta Avenue
Newton, NJ 07860
USA
Tel: 973-579-7227
Fax: 973-300-3600
www.thorlabs.com
www.thorlabs.us (West Coast)
Email: sales@thorlabs.com
Support: techsupport@thorlabs.com

Europe
Thorlabs GmbH
Hans-Böckler-Str. 6
85221 Dachau
Germany
Tel: +49-(0)8131-5956-0
Fax: +49-(0)8131-5956-99
www.thorlabs.de
Email: europe@thorlabs.com

France
Thorlabs SAS
109, rue des Côtes
78600 Maisons-Laffitte
France
Tel: +33 (0) 970 444 844
Fax: +33 (0) 825 744 800
www.thorlabs.com
Email: sales.fr@thorlabs.com

UK and Ireland
Thorlabs Ltd.
1 Saint Thomas Place, Ely
Cambridgeshire CB7 4EX
Great Britain
Tel: +44 (0)1353-654440
Fax: +44 (0)1353-654444
www.thorlabs.com
Email: sales.uk@thorlabs.com
Support: techsupport.uk@thorlabs.com

Scandinavia
Thorlabs Sweden AB
Mölndalsvägen 3
412 63 Göteborg
Sweden
Tel: +46-31-733-30-00
Fax: +46-31-703-40-45
www.thorlabs.com
Email: scandinavia@thorlabs.com

Japan
Thorlabs Japan, Inc.
Higashi-Ikebukuro Q Building, 1F
2-23-2, Higashi-Ikebukuro,
Toshima-ku, Tokyo 170-0013
Japan
Tel: +81-3-5979-8889
Fax: +81-3-5979-7285
www.thorlabs.jp
Email: sales@thorlabs.jp

China
Thorlabs China
Room A101, No. 100
Lane 2891, South Qilianshan Road
Putuo District
Shanghai
China
Tel: +86 (0)21-32513486
Fax: +86 (0)21-32513480
www.thorlabs.hk
Email: chinasales@thorlabs.com
AOKIT1, AOKIT2, AOKIT3, AOKIT4 Series
Adaptive Optics Kit

Software Guide
Table of Contents

Chapter 1  Overview ............................................................................................................................... 4
  1.1. The AOSystem .......................................................................................................................... 4
      1.1.1. Other files ....................................................................................................................... 4
  1.2. Deformable Mirror (DM) ....................................................................................................... 4
  1.3. Shack-Hartmann Wavefront Sensor (WS) ......................................................................... 5

Chapter 2  Header Files .......................................................................................................................... 6
  2.1. Required Included Files ....................................................................................................... 6
  2.2. Optional Included Files for .NET .......................................................................................... 6

Chapter 3  AOSystem API Data Structure (AOSystemData) ............................................................ 7
  3.1. The AOSystem ........................................................................................................................ 7
      3.1.1. AOSystem.DLL Internal Memory Used ....................................................................... 7
      3.1.2. End User Application’s Memory Used ....................................................................... 8
  3.2. The AOSystemData Structure ............................................................................................ 9

Chapter 4  AOSystem API Functions .................................................................................................. 16
  4.1. General functions (AOS_) ..................................................................................................... 16
      4.1.1. AOS_Initialize ............................................................................................................... 16
      4.1.2. AOS_Close ..................................................................................................................... 18
      4.1.3. AOS_GetInfoOnDevices ............................................................................................... 19
      4.1.4. AOS_GetVarAddress .................................................................................................... 20
      4.1.5. AOS_SetVarAddress ..................................................................................................... 22
      4.1.6. AOS_Calibration ........................................................................................................... 24
      4.1.7. AOS_CalcZernikeWavefront ......................................................................................... 26
      4.1.8. AOS_GetZernikeCoeffsFmWavefront ........................................................................... 27
      4.1.9. AOS_UseSpotDeviations ............................................................................................. 28
      4.1.10. AOS_InitCapturedReference ..................................................................................... 30
  4.2. Deformable Mirror Functions (DM_) .................................................................................... 31
      4.2.1. DM_SetVoltageFrame .................................................................................................. 31
      4.2.2. DM_SetSpatialFrame ................................................................................................... 33
      4.2.3. DM_SetQuadraticCoeffAndMaxV .............................................................................. 35
### 4.3. Shack-Hartmann Wavefront Sensor Functions (SH) .................................................. 37

- 4.3.1. SH_GetMeasurements .......................................................................................... 37
- 4.3.2. SH_SetGainExposure ......................................................................................... 39
- 4.3.3. SH_CalcWavefront ............................................................................................ 41
- 4.3.4. SH2_CalcWavefront .......................................................................................... 42
- 4.3.5. SH_SetSpotsToReference ................................................................................. 44
- 4.3.6. SH_SetCalcSpotsToReference ......................................................................... 45
- 4.3.7. SH_GetSpotReferencePositions ....................................................................... 47
- 4.3.8. SH_CalcSpotDeviations ................................................................................... 49
- 4.3.9. SH_GetSpotCentroids ....................................................................................... 51
- 4.3.11. SH_CalcCorrection ........................................................................................ 53
- 4.3.12. SH_GetPupil .................................................................................................... 55
- 4.3.13. SH_SetPupil ..................................................................................................... 57
- 4.3.14. SH_SetReference ............................................................................................ 59
- 4.3.15. SH_GetReference ........................................................................................... 60
- 4.3.16. Function Parameters ...................................................................................... 60

---

**Chapter 5**  
Example Code ........................................................................................................ 61

**Chapter 6**  
Regulatory .................................................................................................................. 62

**Chapter 7**  
Thorlabs Worldwide Contacts .................................................................................. 63
Chapter 1  Overview

The Adaptive Optics Kit (AOKit) comes with software which will allow you to access and control the Deformable Mirror (DM) and Shack-Hartmann Wavefront Sensor (WS). This document is written for developers requiring access to the functions needed.

The following table details the AOKits that use this software package.

| Item #     | Description                                                                 |
|------------|                                                                            |
| AOK1-UM01  | Adaptive Optics Toolkit with Gold Coated 140 Actuator DM                    |
| AOK1-UP01  | Adaptive Optics Toolkit with Aluminum 140 Actuator DM                       |
| AOK2-UM01  | Adaptive Optics Toolkit with Gold Coated 32 Actuator DM                     |
| AOK2-UP01  | Adaptive Optics Toolkit with Aluminum Coated 32 Actuator DM                 |
| AOK3-UM01  | Adaptive Optics Kit with Gold Coated 140 Actuator DM with Fast Wavefront Sensor |
| AOK3-UP01  | Adaptive Optics Kit with Aluminum Coated 140 Actuator DM with Fast Wavefront Sensor |
| AOK4-UM01  | Adaptive Optics Kit with Gold Coated 32 Actuator DM with Fast Wavefront Sensor |
| AOK4-UP01  | Adaptive Optics Kit with Aluminum Coated 32 Actuator DM with Fast Wavefront Sensor |

1.1. The AOSystem

The primary focus of the AOKit is the AOSystem.DLL, which is comprised of all of the functions that the user can access in order to operate the DM and the WS. This AOSystem.DLL was written in the ‘C’ programing language on the Windows OS. The AOSystem.DLL will work with an MFC or a .NET application.

There are two programs that ship with the AOKit: the interactive program called AOKit2, which is an MDI application in which the user can manipulate the DM and the WS, and the AOSysDemoApp, which is an example of how to call most of the functions inside the AOSystem API and comes with documented source code. The demo program (AOSysDemoApp) accompanying the AOSystem.DLL was written in VS 2005.NET C++. Since the nature of the .NET environment lends itself to being able to swap between different languages easily, there should be only minor changes involved to use the AOSystem.DLL with VB.NET or C#.NET.

1.1.1. Other files

The AOKit program ships with other supporting files. These other files should reside with the AOSystem.DLL in your program’s executable directory so that your program will be able to load and access the AOSystem properly:

- CIUsbLib.dll – Needed to access the USB-based DM.
- Interop.CIUsbLib.1.0.dll – A .NET component used to enable calls from .NET to older COM type programs.
- SurfacePlot.dll – A Thorlabs’ User Interface control used to plot the surface profile of the DM and the WS as determined from an inputted set of Zernike coefficients.

1.2. Deformable Mirror (DM)

The DM is capable of reconfiguring its surface topography to cancel the on- and off-axis monochromatic aberrations in the incident wavefront, which would otherwise result in a blurred image or degraded focus spot. The deformable mirrors used in the various AOKits come in two sizes, a 12 x 12 (140 electrostatic actuator) version or a 6 x 6 (32 electrostatic actuator) version. The DM has a broadband enhanced aluminum or gold
coating and is capable of operating at approximately 4 kHz. The amount of voltage applied to each actuator
determines its displacement and ultimately controls the profile of the surface of the DM membrane. Since all
actuators must be set at one time, the API only has functions that work with the grid as a whole. The API
functions that perform the actual manipulation of the deformable mirror can only write to the DM. Presently, there
is no way to read back the status of the actuators from the DM, which is only a writable device.

Please note that all of the specific function calls to the Deformable Mirror are preceded with 'DM_'.

1.3. Shack-Hartmann Wavefront Sensor (WS)

The Shack-Hartmann wavefront sensor allows the user to take high-speed measurements of the wavefront shape
and intensity distribution of beams. This is done by analyzing the location and intensity of spots (spotfield) formed
by imaging a beam of light onto a CCD array with a microlens array.

With the WS, it is possible to dynamically optimize the wavefronts of laser sources, characterize the wavefront
distortion caused by optical components, and provide real-time feedback for the control of adaptive optics.

If a planar wavefront is incident on a WS, the light transmitted through a microlens array and imaged on a CCD
sensor will form a regular pattern of bright spots. If, however, the wavefront is distorted, the light imaged on the
CCD sensor will consist of some regularly spaced spots mixed with displaced spots and missing spots. This
information can be used to calculate the shape of the wavefront that was incident on the microlens array. The
Shack-Hartmann type wavefront sensor can be used to monitor the wavefront in real time and use that
information to flatten or adjust the wavefront with a deformable mirror.

Please note that all of the specific function calls to the Shack-Hartmann are preceded with 'SH_'.
Chapter 2     Header Files

2.1. Required Included Files

This header file must be included at the top of the user's program in order to access the functions in the AOSystem.DLL:

    #include "AOSystem_DataStructures.h"

2.2. Optional Included Files for .NET

Rather than have to setup the declaration for all of the functions in to AOSystem.DLL, this is a sample VS2005.NET header file and could be included at the top of the user's program.

    #include "AOSystem_DllImports.h"

This namespace should be included at the top of your program:

    using namespace AOSystem_DllImports;
Chapter 3  AOSystem API Data Structure (AOSystemData)

3.1. The AOSystem

The AOSYSTEM.DLL application library (see the left box in Figure 1) represents the heart of the whole AOKit system. This library contains all of the functionality that is used to write to the Deformable Mirror and to read from the Shack-Hartmann Wavefront Sensor.

This AOSYSTEM.DLL will be accessed by the “End User Application” (right box in Figure 1), which will typically be a user’s application written in C++, C# or VB. The demo program included with this API, called AOSysDemoApp, highlights the most frequently used function calls. This program makes function requests to the AOSYSTEM.DLL and gets the returned results. On the surface, this is how all the programs will “talk” to this .DLL.

The core part of this DLL is a versatile memory structure called AOSystemData, which is comprised of some smaller structures. The set of user assignable pointers in this memory is the structure’s key advantage (center box in both Figure 1 and Figure 2). A more in-depth description of what happens internally is depicted by the two figures that follow.

The set of user assignable pointers in this memory is the structure’s key advantage (center box in both Figure 1 and Figure 2).

3.1.1. AOSystem.DLL Internal Memory Used

In C++, as with many other computer languages, the AOSYSTEM.DLL will assign its own internal memory, the “AO System Data Memory”, and then setup the “Data Index AOSystemData” to access it. This “Data Index AOSystemData” internally to the AOSYSTEM.DLL is called ‘aoSystemData’ and this is what is used for all internal calls and operations. (See Figure 1)

![Figure 1  The AOSystem API Data Structure](image-url)
3.1.2. End User Application’s Memory Used

In some languages, like Visual Basic for instance, the user does not have the ability to modify memory in the DLL. Since this would seem to cause a conflict, the VB programmer can instead allocate their AOSystemData memory structure from within their VB program and send a pointer of this memory to the AOSYSTEM.DLL. This will assign the Data Index AOSystemData to the user’s memory. The AOSystemData structure will then point to the user’s memory structure instead, thereby allowing VB programmers to work with this memory structure and have the AOSYSTEM.DLL use this memory for its operations. (See Figure 2)

![Diagram](image_url)

**Figure 2** The AOSystem API Data Structure
3.2. The AOSystmeData Structure

#include "WFS_Drv.h"

// These are the return values for the AOS_Initialize call.
#define DEVICES_ALREADY_INITIALIZED -1 //The AOSystem has already been initialized.
#define DEVICE_NONE 0x0 //No devices were found.
#define DEVICE_DM_FOUND 0x1 //A Deformable Mirror (DM) was found.
#define DEVICE_SH_FOUND 0x2 //The Shack-Hartmann (SH) was found.
#define DEVICE_MINI_DM_FOUND 0x4 //The Mini Deformable Mirror (DM) was found.
#define DEVICE_WFS10_FOUND 0x8 //The WFS10 fast WFS was found.
#define DEVICE_OFFSET_WFS10 (0x00100) //device IDs of WFS10 instruments start at
//256 decimal

// These are the return values for the AOS_Close call.
#define SHUTDOWN_OK 0 //The AOSystem has already been shutdown properly.
#define SHUTDOWN_ERROR 1 //The AOSystem has encountered an error during shutdown.

// These are used as the first parameter in the GetVarAddress & SetVarAddress calls.
#define VAR_DM_DESIRED 1
#define VAR_DM_ACTUAL 2
#define VAR_DM_VOLTAGE 3
#define VAR_DM_ALTERNATE 4
#define VAR_DM_FIT 5
#define VAR_DM_COEFF 6
#define VAR_DM_MAP 7 //not used 2009-03-06
#define VAR_DM_CELLMASK 8
#define VAR_IMAGEBUFFER 9
#define VAR_SVD_M 0
#define VAR_SVD_N 11
#define VAR_SVD_U 12
#define VAR_SVD_V 13
#define VAR_SVD_VT 14
#define VAR_SVD_R 15
#define VAR_SVD_R2 16
#define VAR_SVD_R2INV 17
#define VAR_SVD_W 18
#define VAR_SVD_RTMP 19
#define VAR_SVD_ORIG 20
#define VAR_SPOTINCLUSION 21
#define VAR_DM_COMMAND 22
#define VAR_INSTR 23
#define VAR_INSTR_SETUP 24
#define VAR_SPOTINFO 25
#define VAR_DESIRED 26
#define VAR_ZERNIKE_BASE 27
#define VAR_ZERNIKE_COEF 28
#define VAR_ZERNIKE_ACTUAL 29
#define VAR_ZERNIKE_WAVEFRONT 30
#define VAR_CAPTURED_WAVEFRONT 31
#define VAR_DM_DATA 999
These are the return values for the SH_SetGainExposure call.

#define NO_SETTINGS_CHANGED 0x0  //Neither setting was adjusted.
#define GAIN_ADJUSTED 0x1  //The Gain "Limit" was adjusted.
#define EXPOS_ADJUSTED 0x2  //The Exposure "Limit" was adjusted.
#define BOTH_SETTINGS_ACCEPTED 0x3 //Both the Gain & Exposure "Limits" were adjusted.
#define EE_SIZE 64  //bytes in camera's EEPROM

Wavefront Sensors' number of pixel resolution settings

#define NUM_WFS_RESOLUTION_SETTINGS 5

AOSystemData structure and supporting structures

ifndef AO_INIT  //Make sure this is only defined once
struct DM_Voltage
{
    double DMMinVoltage;  //The minimum voltage of the DM
    double DMMaxVoltage;  //The maximum voltage of the DM
    double DMStartStroke;  //The start-range voltage of the DM used in calibration
    double DMEndStroke;  //The end-range voltage of the DM used in calibration
    double DMVoltageScale; //Calculation multiplier for nm stroke DM movements-This
    //is a constant multiplier equal to the total digital
    //range of stroke (16384) divided by the max voltage
    //for the DM (300).
    double DMVoltageScaleInv; //Represents inverse of above (1 / DMVoltageScale)
    int DMBaseMin;   //Calculated values not to be changed
    int DMBaseMax;   //Calculated values not to be changed
};
typedef struct
{
    double centroid_x[MAX_SPOTS_Y][MAX_SPOTS_X];  //spot centroid positions in pixels
    double centroid_y[MAX_SPOTS_Y][MAX_SPOTS_X];  //for adaption to MathLab
    double reference_x[MAX_SPOTS_Y][MAX_SPOTS_X];  //measured spot ref pos (x) in pixels
    double reference_y[MAX_SPOTS_Y][MAX_SPOTS_X];  //measured spot ref pos (y) in pixels
    double ref_grid_x[MAX_SPOTS_Y][MAX_SPOTS_X];  //ideal ref grid in pixels
    double ref_grid_y[MAX_SPOTS_Y][MAX_SPOTS_X];  //+1 for adaption to MathLab
    float scale_x_mm[MAX_SPOTS_X];    //for internal use only
    float scale_y_mm[MAX_SPOTS_Y];    //for internal use only
    float diameter_x[MAX_SPOTS_Y][MAX_SPOTS_X];  //spot diameters (x) in pixels
    float diameter_y[MAX_SPOTS_Y][MAX_SPOTS_X];  //spot diameters (y) in pixels
    float diameter[MAX_SPOTS_Y][MAX_SPOTS_X];  //geometric mean diameter in pixels
    float peak_level[MAX_SPOTS_Y][MAX_SPOTS_X];  //max intensity level in spot area
    float intensity[MAX_SPOTS_Y][MAX_SPOTS_X];  //sum intensity level in spot area
    float deviation_x[MAX_SPOTS_Y][MAX_SPOTS_X];  //measured deviation =
    //spot centroids, reference = pixels
    float deviation_y[MAX_SPOTS_Y][MAX_SPOTS_X];  //measured deviation =
    //spot centroids, reference = pixels
    float devdiff_x[MAX_SPOTS_Y][MAX_SPOTS_X];  //deviation difference
    float devdiff_y[MAX_SPOTS_Y][MAX_SPOTS_X];  //deviation difference
    float deviation_rec_x[MAX_SPOTS_Y][MAX_SPOTS_X];  //reconstructed
    float deviation_rec_y[MAX_SPOTS_Y][MAX_SPOTS_X];  //reconstructed
    float deviation_diff_x[MAX_SPOTS_Y][MAX_SPOTS_X];  //deviation difference
    float deviation_diff_y[MAX_SPOTS_Y][MAX_SPOTS_X];  //deviation difference
    float wavefront[MAX_SPOTS_Y][MAX_SPOTS_X];  //absolute wavefront results in µm

float wavefront_waves[MAX_SPOTS_Y][MAX_SPOTS_X];  //wavefront results in waves (lambda dept)
float wavefront_flip[MAX_SPOTS_X][MAX_SPOTS_Y]; //flipped indices, only
        //used for 3D display
float wf_scale[2][2];  //for scale of 3D graph
float sum_intensity;  //data related to detected spots only
float mean_intensity;
float weighted_tilt_x;  //not used, for simulations only
float weighted_tilt_y;  //not used, for simulations only
float wf_min;  //minimum wavefront value in µm
float wf_max;  //maximum wavefront value in µm
float wf_diff;  //peak-peak of wavefront in µm
float wf_mean;  //mean wavefront level
float wf_rms;  //RMS value of wavefront
float weighted_wf_rms;  //RMS of wavefront, weighted to individual spot intensity
float wf_min_waves;  //same in waves (lambda dependent)
float wf_max_waves;
float wf_diff_waves;
float wf_mean_waves;
float wf_rms_waves;
float weighted_wf_rms_waves;

float diameter_min;  //min diameter of all valid spots
float diameter_max;  //max diameter of all valid spots
float diameter_mean;  //mean diameter of all valid spots
float spots_centroid_x;  //x,y centroid of an individual spot in pixel
float spots_centroid_y;
float spots_diameter_x;  //x,y diameter of an individual spot in pixel
float spots_diameter_y;
float spots_centroid_x_mm;  //same in mm
float spots_centroid_y_mm;
float spots_diameter_x_mm;
float spots_diameter_y_mm;

float steepness_x;  //x,y steepness of the deviation_xy values, which indicates
float steepness_y;  //a defocus term which can be removed optionally mean, offset,
float mean_x;  //ns relate also to this linear fit to approximate the
float mean_y;  //approximate the steepness of deviation_xy values
float offset_x;
float offset_y;
float ns_x;
float ns_y;

int spots_x;  //max (x) spots according to selected image size
int spots_y;  //max (y) spots according to selected image size
int center_spot_x;  //index of center spot (x)
int center_spot_y;  //index of center spot (y)

float mean_dev_x;  //added June 30, 2010 as part of new 'cancel_wf_til' algorithm.
float mean_dev_y;  //added June 30, 2010 as part of new 'cancel_wf_til' algorithm.
// Added to support new fast WFS and new Zernike functionality

double ref_grid_center_x_pix;  // location of ref grid center in pixels from sensor origin.  
  // Calculate after MLA is selected

double ref_grid_center_y_pix;  // location of ref grid center in pixels from sensor origin 
  // Calculate after MLA is selected

double captured_reference_x[MAX_SPOTS_Y+1][MAX_SPOTS_X+1];  // measured ref (x) in pixels 
  // Calculate after MLA is selected

double captured_reference_y[MAX_SPOTS_Y+1][MAX_SPOTS_X+1];  // measured ref (y) in pixels
  // Calculate after MLA is selected

} spotinfo_t;

typedef struct
{

double wf_lambda_nm;  // wavelength in nm

double expos_min;    // min exposure time in ms, read from camera

double expos_max;    // max exposure time in ms, read from camera

double expos_incr;   // exposure time increment in ms, read from camera

double expos_set;    // desired exposure time, user input

double expos_act;    // actual exposure time, rounded by camera

double master_gain;  // electrical gain

double pupil_dia_x_mm;  // x, y pupil diameter in mm

double pupil_dia_y_mm;

double pupil_center_x_mm; // x, y pupil center position in mm

double pupil_center_y_mm;

int wf_unit;  // WF_UNIT_UM = 0, WF_UNIT_WAVES = 1

int graph_select;  // internal use in GUI

int setup_select;

int cam_resol_idx;  // index 0-4 of pre-defined ROI, cam_xpixel[] = {1280, 1024, 
  // 768, 512, 320}; cam_ypixel[] = {1024, 1024, 768, 512, 320};

int average_cnt;  // number of camera image averages

int cam_noise_level_auto;  // 1 = automatic (flexible) noise level in spot detection

int cam_noise_level;  // noise level in digits, intensities below are cut (set to 0)

int expos_auto;    // 0 = manual, 1 = auto exposure

int black_level_auto;  // not available for selected camera

int black_level_offset;  // target black level setting, sent to camera

int disable_calc_extra_data;  // forces calc of extra statistical data if not disabled (=0)

int disable_graphics;  // 1 disables all graphics for increased speed

int sim_tilt;  // for internal simulation, not used

int sim_defocus;

int sim_cylinder;

int cancel_tilt;  // not used

int cancel_defocus;  // not used

int pixels_per_spot;  // cam pixels per lenslet spot, 41 for WFS150

int pupil_circular;  // = 1 forces a circular pupil

int use_intens_centroid;  // use beam centroid as Zernike center

int use_intens_dia;  // use beam diameter as Zernike diameter

int wf_calc_in_pupil;  // limit wavefront calculation and display to within pupil

int cancel_wf_tilt;  // = 1 cancels average wavefront tilt, 
  // accessible by user via setup panel

int zernike_orders_auto;  // = 1 calculates Zernike terms as many as possible 
  // (dependent on detected spot count)

int zernike_orders;  // max calculated zernike order

int fourier_order;  // max recognised Fourier order, 2, 4 or 6

int ref_idx;  // used wavefront reference, WFS_REF_INTERNAL=0, WFS_REF_USER=1

int wavefront_type;  // 0 = measured (WAVEFRONT_MEAS), 1 = reconstructed 
  // (WAVEFRONT_REC), 2 = difference of both (WAVEFRONT_DIFF)
Added for new fast WFS functions

```c
    double master_gain_max;
    double master_gain_min;
    double pupil_dia_x_pix;  // x,y pupil diameter in pixels
    double pupil_dia_y_pix;
    double pupil_center_x_pix; // x,y pupil center position in pixels, // from camera's (0,0) origin.
    double pupil_center_y_pix;
```

Data type definitions

```c
typedef struct
{
    double roc_set_mm;  // target radius of curvature during calibration
    float center_spot_offset_x; // calibration value, +- pixel from image center
    float center_spot_offset_y; // calibration value
    float lenslet_f_um;  // calibration value, lenslet focal length
    unsigned int status;  // single bits indicate errors and warnings
    int selected_dev_id;  // id to identify the selected WFS camera
    int stop_pressed;    // = 1 continuous measurement stopped, finalize remaining calculations and drawings
    int reg_cal_key;   // internal use
    int bytes_per_row;  // bytes per row in actual image buffer
    int rows;    // rows within actual image size
    int cols;    // columns within actual image size
    int user_ref_valid;  // = 1 user reference spots data are valid
    int reset_roll_avg;  // = 1 forces restart of rolling averages
    int do_zernike_preparation; // = 1 prepares least square fit data, req when pupil is changed
    int do_spherical_reference; // = 1 during a spherical reference run

    char version_cam_driver[WFS_BUFFER_SIZE];    // version of camera driver (IDS)
    char version_wfs_driver[WFS_BUFFER_SIZE];    // version of WFS driver (Thorlabs)
    char ee_string[EE_SIZE+1];    // string read from camera’s EEPROM, contains SN and calibration data
    char ee_string_backup[EE_SIZE+1];  // backup stored in registry
    char instrument_name[WFS_BUFFER_SIZE];    // "WFS150" or "WFS150C"
    char serial_number_cam_dll[WFS_BUFFER_SIZE];    // cam SN from camera driver
    char serial_number_cam_ee[WFS_BUFFER_SIZE];    // cam SN from EEPROM, must be same!
    char serial_number_wfs[WFS_BUFFER_SIZE];    // Thorlabs SN of WFS
```
JECTION_API

Chapter 3: AOSystem Data Structure (AOSystemData)

// Added for new fast WFS functions

char manufacturer_name[WFS_BUFFER_SIZE];  //Thorlabs
char mla_name[WFS_BUFFER_SIZE];
int mla_count;
int selected_mla;
intrl selected_mla_index;
int cam_x_resolution[NUM_WFS_RESOLUTION_SETTINGS];
int cam_y_resolution[NUM_WFS_RESOLUTION_SETTINGS];
unsigned long handle;                  //Handle for WFS
double cam_pitch_um;
double lenslet_pitch_um;
double grd_corr_0;
double grd_corr_45;

//

} instr_t;

struct AOSystemData
{

double *dDMDesired;   //DM: The desired actuators array setting (in nm)
double *dDMActual;   //DM: The actual actuator array setting (in nm)
double *dDMFit;    //DM: Future use: Array of desired actuators
                    // settings minus the actual setting achieved

double *dDMCoeff;    //DM: These are the 3 coefficients found
                    // with the deformable mirror documentation

double **svd_u;       //SH: singular value decomposition- used internally

double **svd_v;       //SH: singular value decomposition- used internally

double **svd_r;       //SH: singular value decomposition- used internally

double **svd_r2;      //SH: singular value decomposition- used internally

double **svd_r2inv;   //SH: singular value decomposition- used internally

double **svd_w;       //SH: singular value decomposition- used internally

double **svd_rtmp;    //SH: singular value decomposition- used internally

double **svd_orig;    //SH: singular value decomposition- used internally

double **svd_grwdebug; // Used to debug calibration files

double *dDMCommand;   //Holds the inverse matrix multiplication

double *dDesired;

double *dZernikeValue;

double *dZernikeWavefront; // Used to calculate the Zernike Polynomial
                    // wavefront surface array

double **sh_spots_dx_pix; // Used to store the delta X of the Zernike after
                    // a call to AOS_CalcZernikeWavefront

double **sh_spots_dy_pix; // Used to store the delta Y of the Zernike after
                    // a call to AOS_CalcZernikeWavefront

double *dCapturedWavefrontValue; // The captured wavefront values (in nm)

float SH_Threshold_Off;   //SH: Minimum historicity threshold
float SH_Threshold_On;    //SH: Maximum historicity threshold
```c
int *iDMVoltage;    // DM: Voltage array used to set all actuators. This is voltage * the DMVoltageScale (see page 10)
int *iDMAlternate;  // DM: Used to extrapolate edge actuators values
int *uiDMCellMask;  // The is the mask of what actuators are used or not
int *svd_m;        // SH: singular value decomposition - used internally
int *svd_n;        // SH: Determinates if a particule SH spot will be
int *iSpotInclusion; // included based on the the minimum and maximum
int iCalImageAverages; // Value used to set # images used during AO calibration
int *iZernikeBase;  // Used as base values in the EditControlArray
char *ImageBuffer;  // Used to point to the actual image obtained
instr_t *instr;     // SH: Ptr to hardware info of the SH camera
instr_setup_t *instr_setup; // SH: Ptr to instrument structure for the SH cam
spotinfo_t *spotinfo; // SH: Ptr to SH spot data structure.
struct DM_Voltage Volt; // DM: Pointer to DM voltage variables

#define AO_INIT 1   // This will stop a redefinition of this structure
#endif
```
Chapter 4 AOSystem API Functions

4.1. General functions (AOS_)

4.1.1. AOS.Initialize

This function is used to initialize the AOSystem and should be called before most other calls are made. The only exception to this is if the user wants to use a locally defined aoSystemData structure (for the complete aoSystemData structure, see section 3.1, page 7) in which case they would have to first call the AOS_SetVarAddress function (see section 4.1.5, page 22) to set a pointer to their locally initialized aoSystemData structure. Both the DM and the WS devices will be initialized so that they may be controlled. All control for both of these devices goes through the AOSystem.dll.

Declaration

    int AOS.Initialize(int boardNum);

Function Parameters

    int boardNum

The board number of the DM found in the system. It takes the value of the board with the first starting at an index of 0 and subsequent boards being at 1, 2, etc.

Note: Presently, this parameter is not used and is reserved for future use. Use the default value 0.

AOSystemData Parameters Used

Incoming

None.

Outgoing

The aoSystemData->Volt structure is filled in with these default values:

    aoSystemData->Volt.DMMaxVoltage = 165.0  //Set to 165 to protect the DM
    aoSystemData->Volt.DMStartStroke = 1000.0
    aoSystemData->Volt.DMEndStroke = 2500.0
    aoSystemData->Volt.DMMinVoltage = 0.0
    aoSystemData->Volt.DMVoltageScale = 54.61333
    aoSystemData->Volt.DMVoltageScaleInv = 1 / aoSystemData->Volt.DMVoltageScale
    aoSystemData->Volt.DMBaseMin = 0
    aoSystemData->Volt.DMBaseMax = ((aoSystemData->Volt.DMMaxVoltage * 16384) / 300)
    aoSystemData->dDMDesired = 0
    aoSystemData->iDMMap = 0
    aoSystemData->dDMCoeff = 0
    aoSystemData->uiDMCellMask = 1
The following WS values are set:

```c
aoSystemData->SH_Threshold_Off = 1000.0;
aoSystemData->SH_Threshold_On = 10000.0;
aoSystemData->instr_setup->master_gain = 1.0;
aoSystemData->instr_setup->expos_set = 0.20;
iZernikeBase = 0
dZernikeValue = 0
dZernikeActual = 0
```

The following WS structures and arrays were initialized based on the SH camera specs:

```c
spotinfo
instr_setup
instr
zerninfo
svd_arrays
```

The whole aoSystemData structure has its pointers updated to point to locations of all the initialized variables.

**Function Return Value**

```c
#define DEVICES_ALREADY_INITIALIZED -1 //The AOSystem was already initialized
#define DEVICE_NONE 0x0 //No devices were found
#define DEVICE_DM_FOUND 0x1 //The Deformable Mirror (DM) was found
#define DEVICE_SH_FOUND 0x2 //The Shack-Hartmann (SH) was found
#define DEVICE_DM_AND_SH_FOUND 0x3 //Both the DM and the SH were found
```

**Example Code**

```c
//Initalize the AOSystem
int retStatus = AOS::AOS_Initialize(0);
switch(retStatus)
{
    case DEVICES_ALREADY_INITIALIZED:
        MessageBox::Show("The AOSystem has already been initialized.");
        break;
    case DEVICE_NONE:
        MessageBox::Show("No devices were found.");
        break;
    case DEVICE_DM_FOUND:
        MessageBox::Show("The Deformable Mirror (DM) was found.");
        break;
    case DEVICE_SH_FOUND:
        MessageBox::Show("The Shack-Hartmann (SH) was found.");
        break;
    case DEVICE_DM_AND_SH_FOUND:
        MessageBox::Show("Both the DM and the SH were found.");
        break;
}
```
4.1.2. AOS_Close

This function will close the AOSystem and should be called to terminate the system and shutdown both the DM and the WS devices. This would typically be the last function called before terminating operations with the DM and the WS devices.

**WARNING**

Not calling this will cause memory leaks.

**Declaration**

```c
int AOS_Close(void);
```

**Function Parameters**

None.

**AOSystemData Parameters Used**

*Incoming*

None.

*Outgoing*

The aoSystemData structure pointers.

**Function Return Value**

```c
#define SHUTDOWN_OK 0 //The AOSystem has already been shutdown properly.
#define SHUTDOWN_ERROR 1 //The AOSystem has encountered an error during shutdown.
```

**Example Code**

```c
//Terminate communication with DM and SH.
int status = AOS::AOS_Close();
if (status == 1)
{
    printf("There was an error calling the AOS_Close() function.");
}
4.1.3. AOS_GetInfoOnDevices

This function will return a pointer to a character array that includes information about the DM and WS devices connected to the computer. This can only be called after the AOS_Initialize function has been called.

Declaration

```c
char* AOS_GetInfoOnDevices();
```

Function Parameters

None.

AOSystemData Parameters Used

incoming

None.

Outgoing

None.

Function Return Value

A pointer to a character array is returned that tells if the DM and the WS are connected and related information about them. For the wave front sensor, the following information is acquired: instrument name, serial number, micro-lens array name. For the deformable mirror the following information is acquired: device name, vendor ID and product ID.

NOTE: This information is used in the AOKit2 "About" Dialog Box.

Example Code

```c
// The char pointer will point to the internal 2K buffer that contains the device
//information. This is delimited by 0xd0xa for each line; therefore, one may want
//to parse this for their own applications.

// Get information on the devices connected.
char* devicesInfo = AOS_GetInfoOnDevices();

printf("Device information: %s", devicesInfo);
```
4.1.4. AOS_GetVarAddress

This function is used to get the memory address of the AOSystemData structure or any of the internal memory variable pointers inside of this structure. The user passes the type of array they are working with as the first parameter. The second parameter is a memory pointer. After this function call, the 2nd parameter will be replaced with the address to the AOSystem.DLL’s memory. The end user’s software would now be able to manipulate the data in memory using the address returned.

See section 3.1 for more.

**Declaration**

```c
void AOS_GetVarAddress(unsigned long var, void **addr);
```

**Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long var</td>
<td>The type of array to be fetched. In the example below, a pointer to the whole aoSystemData structure is returned from the AOSystem.DLL. This is done using the value of VAR_DM_DATA (which is defined as 999). Since the Data Index AOSystemData structure (See section 3.1) includes a set of pointers to memory allocated elsewhere, the user can obtain the memory addresses of included variables and can work with these addresses locally. The list of all of the internal memory variables within the aoSystemData structure can be found in the AOSystem_DataStructures.h file. (See section 3.2, page 9).</td>
</tr>
<tr>
<td>void **addr</td>
<td>A pointer to the address of the Data Index AOSystemData structure or one of its variables.</td>
</tr>
</tbody>
</table>

**AOSystemData Parameters Used**

**Incoming**

None.

**Outgoing**

Fetches the memory address of the chosen variable inside the Data Index AOSystemData structure.

**Function Return Value**

None.
Example Code

```c
// Allocate the pointer to point to the DLL's internal memory structure.
struct AOSSystemData *aoSystemData;
// Get address of AOSSystem.DLL internal memory array used to write to the DM.
// Array of ints of DM profile
AOS::AOS_GetVarAddress(VAR_DM_DATA, (void **)andaoSystemData);

// Using this memory address, fill the actuators with 0 values in preparation
// for writing the array to the DM. This 0 value will cause the actuators not
// to be pulled.
for(int x = 0; x < 144; x++)
aoSystemData->iDMVoltage[x] = 0;
// Show a cross using the Gimbal mounted (v2) mirror file.
// Check the DM documentation for more on the actuator positions.
array<int>^J={138,126,114,102,90,78,66,54,42,30,18,6,137,125,113,101,89,77,
              65,53,41,29,17,5, //vertical center lines
              83,82,81,79,76,75,74,73,72,71,70,69,68,67,64,63,62,61,60};
// horizontal center lines (minus 2 center values)

// Now poke specific actuators on the 12 X 12 grid. This is done by changing
// the values for the individual actuators defined above and then calling the
// function for writing.
for (int i=0; i < J->Length; i++)
aoSystemData->iDMVoltage[J[i]] = 0x3FFF;

// Perform the actual writing to the DM.
AOS::DM_SetVoltageFrame();
```
4.1.5. AOS_SetVarAddress

This function is used to set a locally defined memory address for the AOSystemData structure. The user passes the type of array s/he is working with as the first argument. This function sets the user’s local variable address as a pointer from within the AOSystem.DLL as the 2nd argument. It allows local memory to be manipulated by the user’s software, as with any other local variables.

See section 3.1 for more.

Declaration

```c
int AOS_SetVarAddress(unsigned long var, void *addr);
```

Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long var</td>
<td>The type of array to set. In the example below, the whole AOSystemData structure is set to point to the user’s local AOSystemData structure allocated in the user’s program. This is done using the value of VAR_DM_DATA (which is defined as 999). Since the Data Index AOSystemData structure (See section 3.1) is includes a set of pointers to memory allocated elsewhere, the user can decide to point most variables to any internal memory address they choose as long as they have allocated the space for it. The list of all of the internal memory variables within the aoSystemData structure can be found toward the top of the AOSystem_DataStructures.h file. For more information on the AOSystemData structure, see section 3.2.</td>
</tr>
<tr>
<td>Void *addr</td>
<td>Address of local memory array to set into the Data Index AOSystemData structure.</td>
</tr>
</tbody>
</table>

AOSystemData Parameters Used

**Incoming**

Sets the memory address of the chosen variable inside the Data Index AOSystemData structure.

**Outgoing**

None.

Function Return Value

1 if the operation succeeded, 0 if there was an error.
Example Code

```c
struct AOSystemData aoSystemData;
// Set address of AOSystem.DLL internal memory structure to use that of the
// local structure we have created to write to the DM.
// Array of ints of DM profile
AOS::AOS_SetVarAddress(VAR_DM_DATA, (void *)&aoSystemData);

// Using this memory address, fill the actuators with values in preparation
// for writing the array to the DM.
for(int x = 0; x < 144; x++)
    aoSystemData.iDMVoltage[x] = 0;

// Show a cross using the Gimbal mounted (v2) mirror file.
// Check the DM documentation for more on the actuator positions.
array<int>^J={138, 126, 114, 102, 90, 78, 66, 54, 42, 30, 18, 6, 137, 125, 113, 101, 89, 77,
              65, 53, 41, 29, 17, 5, // vertical center lines
              83, 82, 81, 79, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 64, 63, 62, 61, 60};
// horizontal center lines (minus 2 center values)

// Now poke specific actuators on the 12 X 12 grid. This is done by changing
// the values for the individual actuators defined above and then calling the
// function for writing.
for (int i=0; i < J->Length; i++)
    aoSystemData.iDMVoltage[J[i]] = 0x3FFF;

// Perform the actual writing to the DM.
AOS::DM_SetVoltageFrame();
```
4.1.6. AOS_Calibration

This function will recalibrate the system and update the AOSystemData structure. This should be called only after the AOS_Initialize and AOS_SetQuadraticCoeffAndMaxV functions. This will initialize the DM actuators to defined values.

This function is also used to set the callback for updating a display after each DM actuator has changed during the calibration. The pointer to the function is passed to the AOSystem.DLL. When the data has changed, this function (shown in the example below as managed .NET code) will be called.

**Declaration**

```c
void AOS_Calibration(SYNCCALLBACK syncDataCallback);
```

**Function Parameters**

- **SYNCCALLBACK syncDataCallback**
  
  If set to NULL, then this will not call a user’s update routine. A “Marshaled” function pointer in .NET callable from this unmanaged .DLL.
  
  For more information on this use, please refer to the sample programs included on the AOKit2 CDROM.

**AOSystemData Parameters Used**

**Incoming**

For more information on these individual variables, please refer to section 3.2

```c
aoSystemData->Volt.DMStartStroke
aoSystemData->Volt.DMEndStroke
aoSystemData->Volt.DMMaxVoltage
aoSystemData->Volt.DMVoltageScale
aoSystemData->instr_setup->master_gain
aoSystemData->instr_setup->expos_set
aoSystemData->SH_Threshold_Off
aoSystemData->SH_Threshold_On
```

**Outgoing**

All the actuators in the following arrays are filled:

```c
aoSystemData->iDMVoltage //For each actuator, voltage * the DMVoltageScale
aoSystemData->uiDMCellMask
aoSystemData->iDMAlternate
```

All data in the following structures will be overwritten and point to calibrated values:

```c
aoSystemData->spotinfo
aoSystemData->svd_arrays
```

**Function Return Value**

None.
Example Code 1

If the user wishes to calibrate the DM without feedback, use the following code as a guide.

```c
// This will perform the system calibration WITHOUT any callback to methods
// in the user’s program.
#define NULL 0
AOS::AOS_Calibration(NULL);
```

Example Code 2

If the user wishes to receive an update after each actuator change during this system calibration, then they must setup a callback. The example below shows how to set up a callback in C++ VS2005.NET.

This code will initialize the callback to the method, SyncDataWindows(), in a .NET program.

```c
typedef void (*SYNCCALLBACK)();

//Setup for callback updates
syncDataWindows = gcnew SyncCallback(this,andAOSysDemoApp::SyncDataWindows);
gcHSyncDataWindows = GCHandle::Alloc(syncDataWindows);

//This will perform the system calibration WITH a callback to method in the
//user’s program.
//This will perform the setup for the callback to the 2nd method below.
void AOSysDemoApp::setupCallback()
{
    IntPtr ip = Marshal::GetFunctionPointerForDelegate(syncDataWindows);
    SYNCCALLBACK cb = static_cast<SYNCCALLBACK>(ip.ToPointer());
    GC::Collect();

    AOS::AOS_Calibration(cb);
    SyncDataWindows(); //call first from here to update windows
}

//This will be the function that is called when the SH’s data (or image)
//has changed and needs to notify the parent application.
void AOSysDemoApp::SyncDataWindows()
{
    MessageBox::Show("Update code goes in this method.\n\n" + "This demo test message will be shown after each actuator has been changed.");
}
```

NOTE: SyncDataWindows() in the above code is the user’s defined function. For a complete listing, please refer to the AOSysDemoApp program included on the AOKit2 CDROM.
4.1.7. **AOS_CalcZernikeWavefront**

This function is used to calculate the Zernike wavefront surface array based on a user defined number of Zernike Coefficients. Zernike Polynomials are an orthogonal set of polynomials that form a basis set used to represent an aberration. For more information on Zernike polynomials, see D Malacara (1992). Optical Shop Testing. 2nd ed. New York: John Wiley and Sons, Inc. p461-472.

**Declaration**

```c
int AOS_CalcZernikeWavefront(int NumCoeff);
```

**Function Parameters**

- **int NumCoeff**
  The number of Zernike coefficients used to calculate the wavefront surface.

**AOSystemData Parameters Used**

**Incoming**

This Nx1 array must be populated with the Zernike coefficients to be used by AOS_CalcZernikeWavefront. The length of this array, N, is passed into AOS_CalcZernikeWavefront; i.e., NumCoeff = N.

```c
aoSystemData->dZernikeValue
```

**Outgoing**

```c
//AOS_CalcZernikeWavefront uses dZernikeValue to calculate the wavefront
//surface and populate the 23 x 23 array dZernikeWavefront.
AOSystemData->dZernikeWavefront
//differential X of Zernike Wavefront represented in a 23 x 23 array
AOSystemData->sh_spots_dx_pix
//differential Y of Zernike Wavefront represented in a 23 x 23 array
AOSystemData->sh_spots_dy_pix
```

**Function Return Value**

The following are the return values:

- **0** The function was executed and the resulting maximum slope of the calculated Zernike function does not exceed the measureable slope of the WFS.
- **1** The function was executed and the resulting maximum slope of the calculated Zernike function exceeds the measureable slope of the WFS.

**Example Code**

```c
//Populate the single dimension array aoSystemData->dZernikeValue with
//‘double’ values such as { 0.0, 0.5, 0.75, 0.0, 0.25};

//In the case above, there were 5 values to pass
int NumCoeff = 5;
//Calculate Zernike Wavefront
AOS::AOS_CalcZernikeWavefront(NumCoeff);
```
4.1.8. **AOS_GetZernikeCoeffsFmWavefront**

This function populates a user specified array with Zernike coefficients measured by the Shack-Hartmann sensor.

**NOTE:** This function will return the latest Zernike coefficients measured by the Shack-Hartmann wave-front sensor.

Also note that the array of floats passed to this function must be allocated to at least 16 since the zernCoefficients[0] is not used in order to keep the correlation of

\[
\text{Zernike Term #1} = \text{zernCoefficients}[1]
\]

and so on.

### Declaration

```c
int AOS_GetZernikeCoeffsFmWavefront(float *zernCoefficients);
```

### Function Parameters

- **float *zernCoefficients**
  - This points to an array of floats to be filled by this function.

### AOSystemData Parameters Used

- **Incoming**
  - None.

- **Outgoing**
  - None.

### Function Return Value

The number of Zernike Coefficients (plus 1 for the unused [0]).

### Example Code

```c
float zernCoeff[16]; //Allow for 16 Zernike Coefficients for now
//The first in this array is not used. Zernike Coeffs will start at [1]

int numCoeffsReturned = AOS::AOS_GetZernikeCoeffsFmWavefront(andzernCoeff[0]);
for (int index=1; index < numCoeffsReturned; index++)
{
    String ^tempName = "Z" + index + ": " + zernCoeff[index]);
    textBox1->AppendText(tempName + Environment::NewLine);
}
```
4.1.9. AOS_UseSpotDeviations

This function is used to convert the Zernike wavefront array into the reference arrays that are used to correct the DM.

NOTE

If this function is used, it MUST be preceded by the function call:

AOS_CalcZernikeWavefront(int NumCoeff);

This function populates two arrays, one of $\Delta x$ shifts and one of $\Delta y$ shifts of the SH spots, see function parameters below. These two arrays are used by AOS_UserSpotDeviations to calculate the new x and y reference spot positions.

The resulting arrays are aoSystemData->spotinfo->reference_xandy, which are the actuator settings used for waveform display, AND aoSystemData->dDesired which is the desired DM wavefront.

Declaration

```c
void AOS_UseSpotDeviations();
```

Function Parameters

None.

AOSystemData Parameters Used:

**Incoming**

- `aoSystemData->sh_spots_dx_pix` //X deviations of WS spots (pixels) of Zernike
  //Wavefront represented in a 23 x 23 array
- `aoSystemData->sh_spots_dy_pix` //Y deviations of WS spots (pixels) of Zernike
  //Wavefront represented in a 23 x 23 array

**Outgoing**

- `aoSystemData->spotinfo->reference_x` //WS spot reference position X (in pixels)
- `aoSystemData->spotinfo->reference_y` //WS spot reference position Y (in pixels)
- `aoSystemData->dDesired` //Desired reference wavefront used for
  //displaying in AOKit2

Function Return Value

None.
Example Code

```c
// Populate the single dimension array aoSystemData -> dZernikeValue with `double`
// values such as {0.0, 0.5, 0.75, 0.0, 0.25};

// In the case above, there were 5 values to pass
int NumCoeff = 5;

// Calculate Zernike Wavefront
AOS::AOS_CalcZernikeWavefront(NumCoeff);

// Now set the reference_x and reference_y to represent the calc Zernike wavefront.
AOS::UseSpotDeviations();

// The variables below now contain the reference XandY representative of the
// Zernike wavefront, and the displayed reference wavefront:
aoSystemData->spotinfo->reference_x // SH spot reference position X (in pixels)
aoSystemData->spotinfo->reference_y // SH spot reference position Y (in pixels)
aoSystemData->dDesired // Desired reference wavefront used for
// displaying in AOKit2
```
4.1.10. AOS_InitCapturedReference

This function copies aoSystemData->spotinfo->ref_grid_x[ ][ ] and aoSystemData->spotinfo->ref_grid_y[ ][ ] to aoSystemData->spotinfo->captured_reference_x[ ][ ] and aoSystemData->spotinfo->captured_reference_y[ ][ ], respectively. These parameters are defined in structure spotinfo_t. Also, the values of aoSystemData->dCapturedWavefrontValue[ ] are set to zeros. dCapturedWavefrontValue is used to store the captured wavefront in nm.

Declarations

None.

Function Parameters

| void AOS_InitCapturedReference(); |

AOSystemData Parameters Used

**Incoming**

- **CalcRefSpots_x[ ]**
  - This parameter contains a two-dimensional array of float containing the actual reference X spot positions in pixels.
  - The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].
  - Note: First array index is the spot number in Y, second index the spot number in X direction.

- **CalcRefSpots_y[ ]**
  - This parameter contains a two-dimensional array of float containing the actual reference Y spot positions in pixels.
  - The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].
  - Note: First array index is the spot number in Y, second index the spot number in X direction.

Function Return Value

None

Example Code

| AOS::AOS_InitCapturedReference(); |
4.2. Deformable Mirror Functions (DM_)

4.2.1. DM_SetVoltageFrame

This function is used to set the DM using the array of memory previously set by the user. Before calling this function, the user must first obtain the address of the DM’s array in memory (see AOS_GetVarAddress) and then set each actuator with a value. This function uses the data held in the aoSystemData->iDMVoltage array as the values for the 12 x 12 DM actuators.

NOTE: The maximum value for any one actuator should not exceed 0x4000. Each actuator is NOT set to a voltage, but rather to a displacement (the voltage * the DMVoltageScale). The voltage must not exceed the voltage rating for the individual DM.

**Declaration**

```c
void DM_SetVoltageFrame(void);
```

**Function Parameters**

None.

**AOSystemData Parameters Used:**

**Incoming**

- `aoSystemData->iDMVoltage` //DM:Voltage array used to set all actuators. This is The DM voltage * the DMVoltageScale (see page 10)

**Outgoing**

None.

**Function Return Value**

None.
Example Code

```c
struct AOSystemData *aoSystemData;
//Get address of AOSystem.DLL internal memory array used to write to the DM.
//Array of ints of DM profile
AOS::AOS_GetVarAddress( 999, (void **)andaoSystemData);

//Using this memory address, fill the actuators with values in preparation
//for writing the array to the DM.
for(int x = 0; x < 144; x++)
aoSystemData->iDMVoltage[x] = 0;
//Show a cross for version 2 Gimbal mounted mirror
array<int>^J={138,126,114,102,90,78,66,54,42,30,18,6,137,125,113,101,
89,77,65,53,41,29,17,5, //vertical center lines
83,82,81,80,79,76,75,74,73,72,71,70,69,68,67,64,63,
62,61,60}; //horizontal center lines (minus 2 center values)
//Now poke specific actuators on the 12 X 12 grid.
for (int i=0; i < J->Length; i++)
aoSystemData->iDMVoltage[J[i]] = 0x3FFF;

//Perform the actual writing to the DM.
AOS::DM_SetVoltageFrame();
```
4.2.2. DM_SetSpatialFrame

This function is used to set the DM using the array of memory previously set by the user. Before calling this function, the user must first obtain or set the address of the DM’s array in memory (see function AOS_GetVarAddress in section 4.1.4 on page 20 or AOS_SetVarAddress in section 4.1.5 on page 22). The user will then set each actuator with a value (in nm). Please refer to the example program in section 4.2.1 on page 31 for how this is done. This function calculates the voltage mapping (stored in aoSystemData->iDMVoltage) that most closely approximates the desired spatial profile (stored in aoSystemData->iDMDesired) by using the following equation:

\[ \text{aoSystemData->iDMVoltage}[i] = \text{aoSystemData->Volt.DMVoltageScale} \cdot \frac{-B + \sqrt{B^2 - 4AC - D}}{2A}; \]

where the quadratic coefficients are given by:

- \( A = \text{aoSystemData->dDMCoeff}[0] \),
- \( B = \text{aoSystemData->dDMCoeff}[1] \),
- \( C = \text{aoSystemData->dDMCoeff}[2] \),
- \( D = \text{aoSystemData->dDMDesired}[i] \).

**NOTE**

If this function is called without first calling the DM_SetQuadraticCoeffAndMaxV function, this function will fail, and return the value MSG_NO_QUADRATIC, otherwise it returns MSG_NO_ERROR, see the Function Return Value section below.

### Declaration

```c
int DM_SetSpatialFrame(void);
```

### Function Parameters

None.

### AOSystemData Parameters Used

- `#define MSG_NO_ERROR 0 //No error`
- `#define MSG_NO_QUADRATIC -4 //Used to determine if DM_SetSpatialFrame was called before setting the coefficients.`
Example Code

```cpp
array<double> coeff = gcnew array<double>(3);
coeff[0] = 0.0244;
coeff[2] = 0.0;

// Set the DM's coefficients which can be found on the data sheet that came with
// your DM.
DM_SetQuadraticCoeffAndMaxV(coeff, 225.0);

// This will perform the system calibration without any callback to methods in the
// user's program.
#define NULL 0
AOS::AOS_Calibration(NULL);

// Make changes to the spatial values of the actuators with your own loop (index
// being your index of individual actuators.
aoSystemData->dDMDesired[i]

// Perform the actual conversion and writing to the DM.
int status = DM_SetSpatialFrame();
```
4.2.3. DM_SetQuadraticCoeffAndMaxV

This function is used to set the DM default coefficients which are listed on the data sheet that came with your DM (See Figure 3 on page 36). Since the actuator movements are not linear, these coefficients are required so that the AOSystem can correctly calculate the voltage needed to move a particular actuator the desired distance (Z axis).

---

**NOTE**

Failure to set the DM maxVoltage properly could result in damage to the DM by overdriving the actuators if voltage is set to high. Failure to set these coefficients can cause the DM to underperform by not utilizing the mirror’s full stroke capability. The max voltage and coefficients could be set externally but this function sets a flag to prove the function was called and that these variables were set. The DM_SetSpatialFrame function checks to make sure that this flag is ‘true’ before allowing the movement of the mirror.

---

**Declaration**

```c
void DM_SetQuadraticCoeffAndMaxV(double coeff[], double maxVoltage);
```

**Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>double coeff[]</code></td>
<td>The double array (size of 3) of the DM default coefficients used to set DM default coefficients. These coefficients can be found on the data sheet that came with your DM.</td>
</tr>
<tr>
<td><code>double maxVoltage</code></td>
<td>The maximum voltage setting for the particular DM. NOTE: The default maxVoltage is 165 volts. This is set to minimize the chance of damaging the DM.</td>
</tr>
</tbody>
</table>

**AOSystemData Parameters Used:**

**Incoming**

None.

**Outgoing**

- `aoSystemData->Volt.DMMaxVoltage` //maximum voltage of DM (in volts)
- `aoSystemData->dDMCoeff` //coefficients used and set in this function

**Function Return Value:**

None.
Example Code

```cpp
array<double> coeff = gcnew array<double>(3);
coeff[0] = 0.0244;
coeff[2] = 0.0;
// Set the DM’s coefficients. These are found on the data sheet that came with
// your DM.
DM_SetQuadraticCoeffAndMaxV(coeff, 225.0);
```

// This will perform the system calibration without any callback to methods in the
// user’s program.
#define NULL 0
AOS::AOS_Calibration(NULL);

// Make changes to the spatial values of the actuators with your own loop (index
// being your index of individual actuators.
aoSystemData->dDMDesired[i]

// Perform the actual conversion and writing to the DM.
DM_SetSpatialFrame();

![Deflection Curves](image)

**Figure 3** Deflection curves for BMC’s deformable mirror.
4.3. Shack-Hartmann Wavefront Sensor Functions (SH_)

4.3.1. SH_GetMeasurements

This function is used to read the data from the Shack-Hartmann. This function can be used to set the lower and upper threshold values of the SH. These are used to stop pixel fluctuation when a noise (either through ambient lighting or electrical oscillation) turns on and off a pixel in the SH. These are the limits of our hysteresis band.

**Declaration**

```c
void SH_GetMeasurements(float threshold_low, float threshold_high);
```

**Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float threshold_low</td>
<td>The lower threshold hysteresis setting of the SH.</td>
</tr>
<tr>
<td>Float threshold_high</td>
<td>The higher threshold hysteresis setting of the SH.</td>
</tr>
</tbody>
</table>

**AOSystemData Parameters Used**

**Incoming**

If the flag, `aoSystemData->instr_setup->expos_auto`, is turned on, then the exposure and master gain will get automatically set.

There is an auto noise level flag, `aoSystemData->instr_setup->cam_noise_level_auto`, which will check the noise level, `aoSystemData->instr_setup->cam_noise_level`, is greater than 0 and if so, will cut the noise using this level from the image obtained.

```c
aoSystemData->instr_setup->average_cnt  //number of camera image averages, the
                                         //default is 1
```

**Outgoing**

```c
aoSystemData->ImageBuffer  //Ptr to the actual picture taken from the SH
aoSystemData->instr->bytes_per_row  //Number of bytes per row of the above image
aoSystemData->instr->rows  //Number of rows of the above image
aoSystemData->instr->cols  //Number of cols of the above image

aoSystemData->instr_setup->expos_set  //the exposure set for the SH camera
aoSystemData->instr_setup->master_gain  //the gain set for the SH camera
```

There is a status bit returned, `aoSystemData->instr->status`, which may contain a mask status of the following:

```c
#define WFS_STATBIT_POWER_TOO_HIGH    0x0004 // power too high (cam saturated)
#define WFS_STATBIT_POWER_TOO_LOW     0x0008 // power too low (low cam digits)
#define WFS_STATBIT_HIGH_AMBIENT_LIGHT 0x0010 // high ambient light
```

There are many other variables in the `aoSystemData->instr_setup` structure that have changed but most are really internal to this system and are not needed by the user.
Function Return Value

None.

Example Code

```c
aoSystemData->SH_Threshold_Off = 2000.0;
aoSystemData->SH_Threshold_On = 10000.0;

AOS::SH_GetMeasurements(aoSystemData->SH_Threshold_Off, aoSystemData->SH_Threshold_On);
```
4.3.2. SH_SetGainExposure

This function is used to set the gain and the exposure settings on the SH. These passed parameters are checked for validity before setting and will generate an error if they are not in the range for the camera.

**Declaration**

```c
int SH_SetGainExposure(double gain, double expos);
```

**Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>double gain</code></td>
<td>The gain setting of the SH.</td>
</tr>
<tr>
<td><code>double expos</code></td>
<td>The exposure setting of the SH. (in ms)</td>
</tr>
</tbody>
</table>

**AOSystemData Parameters Used**

**Incoming**

None.

**Outgoing**

- `aoSystemData->instr_setup->master_gain` //Gain
- `aoSystemData->instr_setup->expos_set` //Desired exposure time (in ms)
- `aoSystemData->instr_setup->expos_act` //Actual exposure time (in ms)

**Function Return Value**

Returns 0, 1, 2, or 3 indicating the status of the two parameters passed. This status indicates whether one, both, or neither of the parameters were adjusted. A zero indicates that the parameter was not set because the new parameter was the same as the old one.

The definitions below can be used to check the return from this function (these are found in the AOSystem_DataStructures.h file):

```c
#define NO_SETTINGS_CHANGED 0x0 //Neither setting was adjusted.
#define GAIN_ADJUSTED 0x1 //The Gain "Limit" was adjusted.
#define EXPOS_ADJUSTED 0x2 //The Exposure "Limit" was adjusted.
#define BOTH_SETTINGS_ACCEPTED 0x3 //Both Gain and Exposure "Limits" adjusted.
```
```cpp
// set the gain to 1.0 and the exposure to 0.20
int retStatus = AOS::SH_SetGainExposure(1.0, 0.20);

switch(retStatus)
{
    case NO_SETTINGS_CHANGED :
        MessageBox::Show("Neither setting was adjusted.");
        break;
    case GAIN_ADJUSTED :
        MessageBox::Show("The Gain \"Limit\" was adjusted.");
        break;
    case EXPOS_ADJUSTED :
        MessageBox::Show("The Exposure \"Limit\" was adjusted.");
        break;
    case BOTH_SETTINGS_ACCEPTED :
        MessageBox::Show("Both the Gain and Exposure \"Limits\" were adjusted.");
        break;
}
```
4.3.3. **SH_CalcWavefront**

This function has been replaced by **SH2_CalcWavefront**.
4.3.4. SH2_CalcWavefront

This function replaces function SH_CalcWavefront in earlier versions of the AOKit DLL. It returns two two-dimensional arrays containing the actual X and Y reference spot positions in pixels.

**NOTE**

Prior to calling this function, it is required to call SH_CalcSpotDeviations.

**Declaration**

```c
int SH2_CalcWavefront(unsigned long handle, int wavefront_type, int limit_to_pupil_interior, float ArrayWavefront[])
```

**Function Parameters**

- **instrumentHandle**
  
  This parameter accepts the Instrument Handle that is set by the AOS_Init function to select the desired instrument driver session. The handle is stored in the instr_t structure, see page xxx.

- **wavefront_type**
  
  This parameter defines the type of wavefront to calculate. Valid settings:
  0  Measured Wavefront
  1  Reconstructed Wavefront based on Zernike coefficients

- **limit_to_pupil_interior**
  
  This parameter defines if the Wavefront should be calculated based on all detected spots or only within the defined pupil. Valid settings:
  0  Calculate Wavefront for all spots
  1  Limit Wavefront to pupil interior

- **ArrayWavefront**
  
  This parameter returns a two-dimensional array of float containing the wavefront data in µm. Passed by reference. The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].

**AOSystemData Parameters**

**Incoming**

- `aoSystemData->instr->handle` // Handle for WFS
- `aoSystemData->instr_setup->wavefront_type` //0 = measured (WAVEFRONT_MEAS), 1 = reconstructed (WAVEFRONT_REC), 2 = difference of both (WAVEFRONT_DIFF)
- `aoSystemData->instr_setup->wf_calc_in_pupil` //limit wavefront calculation and display to within pupil

**Outgoing**

- `aoSystemData->spotinfo->wavefront` //absolute wavefront results in µm
Function Return Value

status

This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

Example Code

```cpp
int err;
//Wavefront types. These are defined within a header file used internally to //AOSystem.dll
#define WAVEFRONT_MEAS (0)
#define WAVEFRONT_REC (1)
#define WAVEFRONT_DIFF (2)
err = AOS::SH2_CalcWavefront(aoSystemData->instr->handle, WAVEFRONT_MEAS, aoSystemData->instr_setup->wf_calc_in_pupil, *aoSystemData->spotinfo->wavefront);
```
4.3.5. SH_SetSpotsToReference

This function sets the reference array from the actual centroid values. It is a direct copy from the centroid X and Y arrays to the reference X and Y arrays.

**Declaration**

```c
int SH_SetSpotsToReference(spotinfo_t *spotinfo);
```

**Function Parameters**

```
spotinfo_t *spotinfo
```

A pointer to the structure of the spot data. This structure consists of (but is not limited to) the following elements: centroid, offset, mean, diameter, deviation, wavefront, and intensity data.

**AOSystemData Parameters Used**

**Incoming**

```
spotinfo->spots_x  //represents the width of the centroid array (int)
spotinfo->spots_y  //represents the height of the centroid array (int)
spotinfo->centroid_x //center spot array x (in pixels)
spotinfo->centroid_y //center spot array y (in pixels)
```

**Outgoing**

```
spotinfo->reference_x //Holds same values as was in centroid_x (in pixels)
spotinfo->reference_y //Holds same values as was in centroid_y (in pixels)
```

**Function Return Value**

Returns int 0 – Not presently used.

**Example Code**

```c
AOS::SH_SetSpotsToReference(aoSystemData->spotinfo);
```
4.3.6. **SH_SetCalcSpotsToReference**

This function copies user calculated spot centroid positions to the User Reference spot positions.

**Declarations**

```c
int SH_SetCalcSpotsToReference(float CalcRefSpots_x[][], float CalcRefSpots_y[][]);
```

**Function Parameters**

- **CalcRefSpots_x[]**
  
  This parameter contains a two-dimensional array of float containing the actual reference X spot positions in pixels. The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].
  
  **Note:** First array index is the spot number in Y, second index the spot number in X direction.

- **CalcRefSpots_y[]**
  
  This parameter contains a two-dimensional array of float containing the actual reference Y spot positions in pixels. The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].
  
  **Note:** First array index is the spot number in Y, second index the spot number in X direction.

**AOSystemData Parameters Used**

- **Incoming**
  
  None.

- **Outgoing**
  
  - CalcRefSpots_x  //center spot array y (in pixels)
  - CalcRefSpots_y  //center spot array y (in pixels)

**Function Return Value**

- **status**
  
  This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.
Example Code

```c
float fTempRef_x[MAX_SPOTS_Y][MAX_SPOTS_X];
float fTempRef_y[MAX_SPOTS_Y][MAX_SPOTS_X];

for (int y = 0; y < ZERN_PLOT_ROWS; y++)
{
    for (int x = 0; x < ZERN_PLOT_COLS; x++)
    {
        fTempRef_x[y][x] = reference_x[y][x];
        fTempRef_y[y][x] = reference_y[y][x];
    }
}
AOS::SH_SetCalcSpotsToReference(fTempRef_x, fTempRef_y);
```
4.3.7. **SH_GetSpotReferencePositions**

This function returns two two-dimensional arrays containing the actual X and Y reference spot positions in pixels.

### Declaration

```c
int SH_GetSpotRefPositions(unsigned long WFS_Handle, int cancelWavefrontTilt,
                           float arrayRefPosition_x[], float arrayRefPosition_y[])
```

### Function Parameters

- **instrumentHandle**: This parameter accepts the Instrument Handle that is set by the AOS_Initialize function to select the desired instrument driver session. The handle is stored in the instr_t structure, see page xxx.
  * WFS_hdl = unsigned long
- **cancelWavefrontTilt**: Internal to SH_GetSpotReferencePositions the wavefront is calculated. This parameter forces the mean spot deviations to be canceled so that the average wavefront tilt will disappear when calculated.
  Valid values:
  - 0 Calculate deviations normal
  - 1 Subtract mean deviation from all spot deviations
- **arrayRefPosX**: This parameter returns a two-dimensional array of float containing the actual reference X spot positions in pixels.
  The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].
  **Note**: First array index is the spot number in Y, second index the spot number in X direction.
- **arrayRefPosY**: This parameter returns a two-dimensional array of float containing the actual reference Y spot positions in pixels.
  The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].
  **Note**: First array index is the spot number in Y, second index the spot number in X direction.
- **status**: This value shows the status code returned by the function call.
  A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

### AOSystemData Parameters Used

**Incoming**

- `aoSystemData->instr->handle` //Handle for WFS
- `aoSystemData->instr_setup->cancel_wf_tilt` //=1 cancels average wavefront tilt, accessible by user via setup panel

**Outgoing**

- `arrayRefPosition_x`
- `arrayRefPosition_y`
### Function Return Value

<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
</tr>
</tbody>
</table>

This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

### Example Code

```c
int err;

float fTempRefX[MAX_SPOTS_Y][MAX_SPOTS_X];
float fTempRefY[MAX_SPOTS_Y][MAX_SPOTS_X];
err = AOS::SH_GetSpotRefPositions(aoSystemData->instr->handle, aoSystemData->instr_setup->cancel_wf_tilt,, *fTempRefX, *fTempRefY);
```
4.3.8. SH_CalcSpotDeviations

This function calculates displacements from their reference positions of the spots on camera CCD array, caused by the incident wavefront on each lenslet of the microlens array. Using this function, the user can calculate the spot deviation from either the internal Zernike reference grid or a user-supplied grid. The internal reference grid is set is stored in the SH camera EPROM, and is the reference for a theoretical flat wavefront incident normal to the SH lenslet array.

NOTE

This function is required in order to calibrate the Shack-Hartmann. During the calibration procedure it is used to measure the effect that each actuator has on the measured wavefront.

Declaration

```c
int SH_CalcSpotDeviations(int ref_idx, spotinfo_t *spotinfo);
```

Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int ref_idx</code></td>
<td>The reference index. It is used to determine the reference spot locations for the deviation calculation. Use <code>ref_idx = 0</code> for: WFS_REF_INTERNAL (Globally defined as 0) - the internal zernike reference grid (alignment of the center spot) OR Use <code>ref_idx = 1</code> for: WFS_REF_USER (Globally defined as 1) – the stored centroids measured from the spotfield.</td>
</tr>
<tr>
<td><code>spotinfo_t *spotinfo</code></td>
<td>A pointer to the structure of the spot data. This structure consists of (but is not limited to) the following elements: centroid, offset, mean, diameter, deviation, wavefront, and intensity data.</td>
</tr>
</tbody>
</table>
AOSystemData Parameters Used

**Incoming**

spotinfo->spots_x  //represents the width of the centroid array (int)
spotinfo->spots_y  //represents the height of the centroid array (int)
spotinfo->centroid_x //center spot array x (pixels)
spotinfo->centroid_y //center spot array y (pixels)

if ref_idx is WFS_REF_INTERNAL:
  spotinfo->ref_grid_x //holds stored centroid x from zernike reference grid (pixels)
  spotinfo->ref_grid_y //holds stored centroid y from zernike reference grid (pixels)

if ref_idx is WFS_REF_USER:
  spotinfo->reference_x //holds stored centroid x from spotfield (pixels)
  spotinfo->reference_y //holds stored centroid y from spotfield (pixels)

**Outgoing**

spotinfo->deviation_x //calculated spot deviation array x (pixels)
spotinfo->deviation_y //calculated spot deviation array y (pixels)

**Function Return Value**

Returns int 0 – Not presently used.

**Example Code**

```c
// This function call is used to show that it precedes the SH_CalcSpotDeviations function.
AOS::SH_GetMeasurements(aoSystemData->SH_Threshold_Off,
                       aoSystemData->SH_Threshold_On);

//Typically, this function call follows the SH_GetMeasurements function above.
AOS::SH_CalcSpotDeviations (WFS_REF_INTERNAL,aoSystemData->spotinfo);
```

NOTE: WFS_REF_INTERNAL is globally defined as 0.
4.3.9. **SH_GetSpotCentroids**

This function returns two two-dimensional arrays containing the centroid X and Y positions in pixels calculated by function SH_GetMeasurements.

### Note

Function **SH_GetMeasurements** is required to run successfully before calculated data can be retrieved.

### Declaration

```c
int SH_GetSpotCentroids (WFS_hdl instrumentHandle, float arrayCentroidX[], float arrayCentroidY[]);
```

### Function Parameters

- **instrumentHandle**: This parameter accepts the Instrument Handle that is set by the AOS_Initialize function to select the desired instrument driver session. The handle is stored in the instr_t structure, see page xxx. * WFS_hdl = unsigned long

- **arrayCentroidX**: This parameter returns a two-dimensional array of float containing the centroid X spot positions in pixels.
  
  The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].

  **Note**: First array index is the spot number in Y, second index the spot number in X direction.

- **arrayCentroidY**: This parameter returns a two-dimensional array of float containing the centroid Y spot positions in pixels.
  
  The required array size is [MAX_SPOTS_Y][MAX_SPOTS_X].

  **Note**: First array index is the spot number in Y, second index the spot number in X direction.

- **status**: This value shows the status code returned by the function call.
  A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

### AOSystemData Parameters Used

#### Incoming

- `aoSystemData->instr->handle`  //Handle for WFS

#### Outgoing

- `aoSystemData->spotinfo->spots_centroid_x`  //x centroids of indiv spots in pXls
- `aoSystemData->spotinfo->spots_centroid_y`  //y centroids of indiv spots in pXls

### Function Return Value

- **status**: This value shows the status code returned by the function call.
  A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.
Example Code

```c
float fTempWFSCentroid_x[MAX_SPOTS_Y][MAX_SPOTS_X];
float fTempWFSCentroid_y[MAX_SPOTS_Y][MAX_SPOTS_X];
err = AOS::SH_GetSpotCentroids(aoSystemData->instr->handle, *fTempWFSCentroid_x,
                            *fTempWFSCentroid_y);
```
4.3.10. SH_SVD_Inverse

This function will calculate the singular value decomposition for the current matrix and inverse if requested.

NOTE: This function uses data pointed to by the "svd_" set of array pointers in the aoSystemData structure.

Declaration

```c
void SH_SVD_Inverse(Boolean bInverse);
```

Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean bInverse</td>
<td>Parameter used to identify which type of singular value decomposition calculation to perform:</td>
</tr>
<tr>
<td></td>
<td>FALSE or 0 – Perform straight Matrix Diagonal Multiply.</td>
</tr>
<tr>
<td></td>
<td>OR</td>
</tr>
<tr>
<td></td>
<td>TRUE or 1 – Perform an Inverse Matrix Diagonal Multiply.</td>
</tr>
</tbody>
</table>

AOSystemData Parameters Used

<table>
<thead>
<tr>
<th>Incoming</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>aoSystemData-&gt;svd_m            //Matrix height</td>
</tr>
<tr>
<td></td>
<td>aoSystemData-&gt;svd_n            //Matrix width</td>
</tr>
<tr>
<td></td>
<td>aoSystemData-&gt;svd_u            //Unitary Matrix</td>
</tr>
<tr>
<td></td>
<td>aoSystemData-&gt;svd_v            //The conjugate transpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outgoing</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aoSystemData-&gt;svd_vt</td>
<td>//Matrix transposed</td>
</tr>
<tr>
<td>if (bInverse == FALSE) aoSystemData-&gt;svd_r2</td>
<td>//The Multiplied Matrix</td>
</tr>
<tr>
<td>else if (bInverse == TRUE) aoSystemData-&gt;svd_r2inv</td>
<td>//The Inverse Multiplied Matrix</td>
</tr>
</tbody>
</table>

Function Return Value

None

Example Code

```c
if (userRequestIsMatrixInverse == true)
    AOS::SH_SVD_Inverse(true);
else
    AOS::SH_SVD_Inverse(false);
```
4.3.11. SH_CalcCorrection

This function calculates and sets the new actuator positions of the DM based on the spotinfo deviation readings from the Shack-Hartmann.

NOTE: This function uses data pointed to by the "svd_" set of array pointers in the aoSystemData structure.

Declaration

```c
void SH_CalcCorrection(double dGain);
```

Function Parameters

- **double dGain**
  
  This is the gain setting used to calculate the desired actuator grid. This is the negative of a number between 1.0 and the maxDisplacement of the actuators where:

  ```c
  double maxDisplacement = aoSystemData->dDMCoeff[0] * 
  Math::Pow(aoSystemData->Volt.DMMaxVoltage, 2) + 
  aoSystemData->dDMCoeff[1] * 
  aoSystemData->Volt.DMMaxVoltage + aoSystemData->dDMCoeff[2];
  ```

AOSystemData Parameters Used

**Incoming**

- `aoSystemData->spotinfo->spots_x`  //represents the width of the centroid array
- `aoSystemData->spotinfo->spots_y`  //represents the height of the centroid array
- `aoSystemData->spotinfo->deviation_x`  //calculated spot deviation array x
- `aoSystemData->spotinfo->deviation_y`  //calculated spot deviation array y
- `aoSystemData->svd_r2inv`  //The Inverse Multiplied Matrix

**Outgoing**

- `aoSystemData->dDMCommand`  //Holds the inverse matrix multiplication
- `aoSystemData->dDMDesired`  //DM: The desired actuators array setting (in nm)
- `aoSystemData->iDMVoltage`  //DM: Voltage array used to set all actuators. This is: voltage * the DMVoltageScale

Function Return Value

None

Example Code

```c
// Calculate new actuator positions.
AOS::SH_CalcCorrection(-dGain);
```
4.3.12. SH_GetPupil

This function returns the actual the pupil position and size.

Declaration

```c
int SH_GetPupil (WFS_hdl instrumentHandle, double *pupilCenterXMm, double *pupilCenterYMm, double *pupilDiameterXMm, double *pupilDiameterYMm);
```

Function Parameters

- **instrumentHandle**: This parameter accepts the Instrument Handle that is set by the AOS_Initialize function to select the desired instrument driver session. The handle is stored in the instr_t structure, see page xxx. * WFS_hdl = unsigned long
- **pupilCenterXMm**: This parameter defines the pupil center X position in mm. The origin is the image center. The valid range is -5.0 to 5.0 mm; although, the region of interest set for the WFS in the AOKit is 3.57 mm x 3.57 mm.
- **pupilCenterYMm**: This parameter defines the pupil center Y position in mm. The origin is the image center. The valid range is -5.0 to 5.0 mm; although, the region of interest set for the WFS in the AOKit is 3.57 mm x 3.57 mm.
- **pupilDiameterXMm**: This parameter defines the pupil X diameter in mm. Valid range: 0.1 to 10.0 mm. Note, the region of interest set for the WFS in the AOKit is 3.57 mm x 3.57 mm.
- **pupilDiameterYMm**: This parameter defines the pupil Y diameter in mm. Valid range: 0.1 to 10.0 mm. Note, the region of interest set for the WFS in the AOKit is 3.57 mm x 3.57 mm.
- **status**: This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

AOSystemData Parameters Used

- **Incoming**
  ```
  aoSystemData->instr->handle //Handle for WFS
  ```

- **Outgoing**
  ```
  pupilCenterXMm
  pupilCenterYMm
  pupilDiameterXMm
  pupilDiameterYMm
  ```
### Function Return Value

**status**

This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

### Example Code

```c
int err;
double dTempPupil_x, dTempPupil_y, dTempPupilDia_x, dTempPupilDia_y;
err = AOS::SH_GetPupil(aoSystemData->instr->handle, &dTempPupil_x, &dTempPupil_y,
                      &dTempPupilDia_x, &dTempPupilDia_y);
```
4.3.13. SH_SetPupil

This function defines the pupil in position and size.

**Declaration**

```c
int SH_SetPupil (WFS_hdl WFS_Handle, double pupilCenterXMm, double pupilCenterYMm, double pupilDiameterXMm, double pupilDiameterYMm);
```

**Function Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_Handle</td>
<td>This parameter accepts the Instrument Handle that is set by the AOS_Initialize function to select the desired instrument driver session. The handle is stored in the instr_t structure, see page xxx. * WFS_hdl = unsigned long.</td>
</tr>
<tr>
<td>pupilCenterXMm</td>
<td>This parameter defines the pupil center X position in mm. The origin is the image center. The valid range is -5.0 to 5.0 mm; although, the region of interest set for the WFS in the AOKit is 3.57 mm x 3.57 mm.</td>
</tr>
<tr>
<td>pupilCenterYMm</td>
<td>This parameter defines the pupil center Y position in mm. The origin is the image center. The valid range is -5.0 to 5.0 mm; although, the region of interest set for the WFS in the AOKit is 3.57 mm x 3.57 mm.</td>
</tr>
<tr>
<td>pupilDiameterXMm</td>
<td>This parameter defines the pupil X diameter in mm. Valid range: 0.1 to 10.0 mm. Note, the region of interest set for the WFS in the AOKit is 3.57 mm x 3.57 mm.</td>
</tr>
<tr>
<td>pupilDiameterYMm</td>
<td>This parameter defines the pupil X diameter in mm. Valid range: 0.1 to 10.0 mm. Note, the region of interest set for the WFS in the AOKit is 3.57 mm x 3.57 mm.</td>
</tr>
<tr>
<td>status</td>
<td>This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.</td>
</tr>
</tbody>
</table>

**AOSystemData Parameters Used**

**Incoming**

- `aoSystemData->instr->handle` //Handle for WFS
- `aoSystemData->instr_setup->pupil_center_x_mm` //x,y pupil center position in mm
- `aoSystemData->instr_setup->pupil_center_y_mm`
- `aoSystemData->instr_setup->pupil_dia_x_mm` //x,y pupil diameter in mm
- `aoSystemData->instr_setup->pupil_dia_y_mm`

**Outgoing**

None.
Function Return Value

**status**

This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

Example Code

```c
int err;
err = AOS::SH_SetPupil(aoSystemData->instr->handle, aoSystemData->instr_setup->pupil_center_x_mm, aoSystemData->instr_setup->pupil_center_y_mm,
aoSystemData->instr_setup->pupil_dia_x_mm, aoSystemData->instr_setup->pupil_dia_y_mm);
```
### 4.3.14. SH_SetReference

This function defines the WFS Reference to either Internal or User (external).

#### Declaration

```c
int SH_SetReference (WFS_hdl instrumentHandle, int referenceIndex);
```

#### Function Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instrumentHandle</td>
<td>This parameter accepts the Instrument Handle that is set by the AOS_Initialize function to select the desired instrument driver session. The handle is stored in the instr_t structure, see page xxx. <em>WFS_hdl</em> = unsigned long</td>
</tr>
<tr>
<td>referenceIndex</td>
<td>This parameter sets the Reference to either Internal or User (external). Valid values: 0: WFS_REF_INTERNAL, 1: WFS_REF_USER. NOTE: these two variables are defined in WFS_Drv.h</td>
</tr>
<tr>
<td>status</td>
<td>This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.</td>
</tr>
</tbody>
</table>

#### AOSystemData Parameters Used

**Incoming**

- `aoSystemData->instr->handle` // Handle for WFS
- `aoSystemData->instr_setup->ref_idx` // Set to WFS_REF_INTERNAL or WFS_REF_USER

**Outgoing**

None.

#### Function Return Value

**status**

This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

#### Example Code

```c
int err;
err = AOS::SH_SetReference(aoSystemData->instr->handle, aoSystemData->instr_setup->ref_idx);
```
4.3.15. SH_GetReference

This function returns the Reference setting of the WFS instrument. This will either be Internal or User (external).

**Declaration**

```c
int SH_GetReference (WFS_hdl instrumentHandle, int *referenceIndex);
```

**4.3.16. Function Parameters**

- **instrumentHandle**
  
  This parameter accepts the Instrument Handle that is set by the AOS.Initialize function to select the desired instrument driver session. The handle is stored in the instr_t structure, see page xxx. * WFS_hdl = unsigned long

- **referenceIndex**
  
  This parameter returns the actual Reference Plane of the WFS instrument.

  Valid Function Function Return Values:

  0 WFS_REF_INTERNAL
  1 WFS_REF_USER

  NOTE: these two variables are defined in WFS_Drv.h

- **status**
  
  This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

**AOSystemData Parameters Used**

**Incoming**

- aoSystemData->instr->handle  //Handle for WFS

**Outgoing**

- referenceIndex  //Passed by reference. Can be aoSystemData->instr_setup->ref_idx

**Function Return Value**

- status

  This value shows the status code returned by the function call. A Function Return Value of zero (0) indicates the function was executed successfully. Other values will be used in future versions of the code.

**Example Code**

```c
int iTempRefIdx;
err = AOS::SH_GetReference(aoSystemData->instr->handle, &iTempRefIdx);
```
Chapter 5  Example Code

Please see the VS 2005.NET Demo Programs for a complete version of example code that accompanies this documentation:

For C++.NET users, see AOSysDemoApp,. - This program references mostly the DM methods used and shows how they may be called in the AOSystem API. It supports a few #defines which allow the program to use different memory structures.

For VB.NET see AOSysVBDemoApp,. – This VB program shows how the user can define their own aoSystemData structure instead of using the memory structure in the AOSystem.DLL.
Chapter 6 Regulatory

As required by the WEEE (Waste Electrical and Electronic Equipment Directive) of the European Community and the corresponding national laws, Thorlabs offers all end users in the EC the possibility to return “end of life” units without incurring disposal charges.

- This offer is valid for Thorlabs electrical and electronic equipment:
- Sold after August 13, 2005
- Marked correspondingly with the crossed out “wheelie bin” logo (see right)
- Sold to a company or institute within the EC
- Currently owned by a company or institute within the EC
- Still complete, not disassembled and not contaminated

As the WEEE directive applies to self-contained operational electrical and electronic products, this end of life take back service does not refer to other Thorlabs products, such as:

- Pure OEM products, that means assemblies to be built into a unit by the user (e.g. OEM laser driver cards)
- Components
- Mechanics and optics
- Left over parts of units disassembled by the user (PCB’s, housings etc.).

If you wish to return a Thorlabs unit for waste recovery, please contact Thorlabs or your nearest dealer for further information.

6.1 Waste Treatment is Your Own Responsibility

If you do not return an “end of life” unit to Thorlabs, you must hand it to a company specialized in waste recovery. Do not dispose of the unit in a litter bin or at a public waste disposal site.

6.2 Ecological Background

It is well known that WEEE pollutes the environment by releasing toxic products during decomposition. The aim of the European RoHS directive is to reduce the content of toxic substances in electronic products in the future.

The intent of the WEEE directive is to enforce the recycling of WEEE. A controlled recycling of end of life products will thereby avoid negative impacts on the environment.
Chapter 7  Thorlabs Worldwide Contacts

USA, Canada, and South America
Thorlabs, Inc.
56 Sparta Avenue
Newton, NJ 07860
USA
Tel: 973-579-7227
Fax: 973-300-3600
www.thorlabs.com
www.thorlabs.us (West Coast)
Email: sales@thorlabs.com
Support: techsupport@thorlabs.com

Europe
Thorlabs GmbH
Hans-Böckler-Str. 6
85221 Dachau
Germany
Tel: +49-(0)8131-5956-0
Fax: +49-(0)8131-5956-99
www.thorlabs.de
Email: europe@thorlabs.com

UK and Ireland
Thorlabs Ltd.
1 Saint Thomas Place, Ely
Cambridgeshire CB7 4EX
Great Britain
Tel: +44 (0)1353-654440
Fax: +44 (0)1353-654444
www.thorlabs.com
Email: sales.uk@thorlabs.com
Support: techsupport.uk@thorlabs.com

France
Thorlabs SAS
109, rue des Côtes
78600 Maisons-Laffitte
France
Tel: +33 (0) 970 444 844
Fax: +33 (0) 825 744 800
www.thorlabs.com
Email: sales.fr@thorlabs.com

Scandinavia
Thorlabs Sweden AB
Möndalsvägen 3
412 63 Göteborg
Sweden
Tel: +46-31-733-30-00
Fax: +46-31-703-40-45
www.thorlabs.com
Email: Scandinavia@thorlabs.com

Japan
Thorlabs Japan, Inc.
Higashi-Ikebukuro Q Building, 1F
2-23-2, Higashi-Ikebukuro,
Toshima-ku, Tokyo 170-0013
Japan
Tel: +81-3-5979-8889
Fax: +81-3-5979-7285
www.thorlabs.jp
Email: sales@thorlabs.jp

China
Thorlabs China
Room A101, No. 100
Lane 2891, South Qilianshan Road
Putuo District
Shanghai
China
Tel: +86 (0)21-60561122
Fax: +86 (0)21-32513480
www.thorlabs.hk
Email: chinasales@thorlabs.com