Advanced OEM Solutions

Software Introduction

OEM-PA/OEM-MC/FMC

Version 1.0

REVISION HISTORY

Date	Rev. No.	Description
2012/05/02		General Edit
2012/05/25		Introduction review
2012/05/25		OEMPASector SW explanation
2012/06/12		Rename software
2012/06/12		Add the last installation kit (driver kit)
2012/06/12		Revision of folder installation
2012/06/12		Introduction review (add Manager SW)
2012/06/12		Add “Tips”
2012/08/06	0.0.12.1	Introduction review
2012/08/06	0.0.12.1	Installer prerequisite about .NET framework 3.5.
2012/09/01	0.0.13.0	Evaluation kit.
2012/09/01	0.0.13.0	Default setup file at connection time.
2012/09/01	0.0.14.0	OEM-PA file options, filter on each cycle, encoder type.
2012/10/20	1.0.4.0	Only one installer is required now.
2012/11/16	1.1.1.0	InstallShield is used for installation. Notepad is automatically associated to “.txt” files.
2013/03/26	1.1.1.0	Support of 64-bit computers.
2013/04/02	1.1.1.0	Rename “Cleaner” to “Manager”, “CustomFilter” to “DefaultConfiguration”.
2013/04/02	1.1.1.0	Add paragraph “Version Compatibility”.
2013/04/02	1.1.2.1	Support of C#.
2013/12/02	1.1.4.1	Update all documentation.
2014/12/12	1.0	General Update

CONTENTS

1. Introduction

This document explains the general architecture of the software package provided to the customer. For all hardware units, AOS provides an Open Source Software Development Kit (SDK) with a very well documented API. Software languages used to build the applications are C++ and C# with driver support tools for LabVIEW, MATLAB & more.

The provided software package contains easy-to-install executables for 32-bit and 64-bit systems:

- 32 bit computer: *AOS OEMPA Kit32 Open Source [version].exe*
- 64 bit computer: *AOS OEMPA Kit64 Open Source [version].exe*
  *As of October, 2015: current [version] = 1.1.5.2

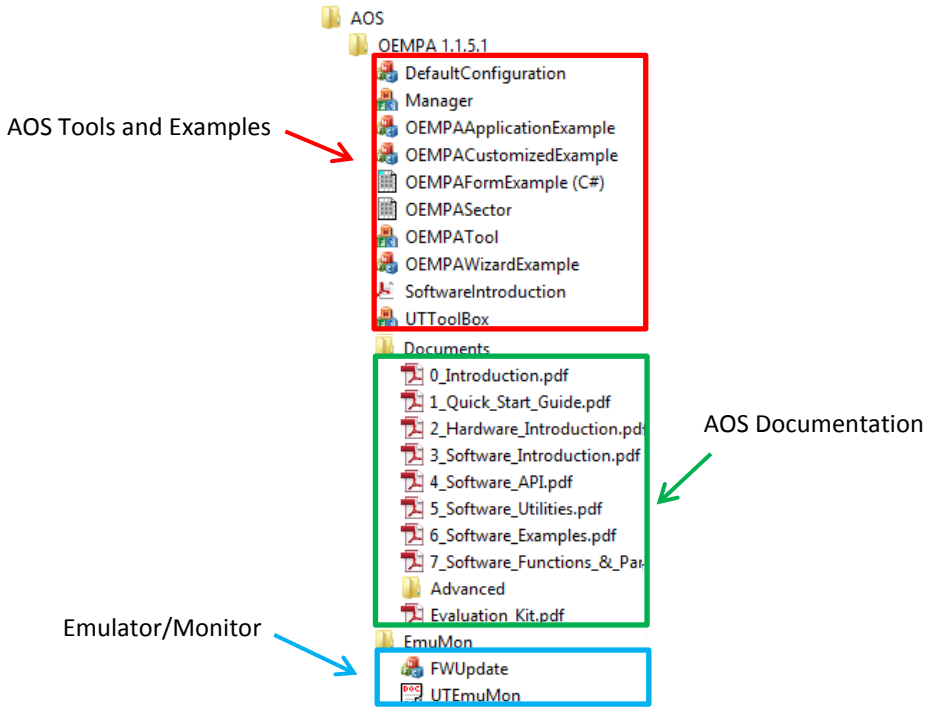
There are also two evaluation software packages, built to evaluate the software without any hardware. In these evaluation kits, *'EmuMon'* is used to emulate the hardware, but only one acquisition data example is included.

- 32 bit computer: *AOS OEMPA Kit32 Evaluation [version].exe*
- 64 bit computer: *AOS OEMPA Kit64 Evaluation [version].exe*

The installer package consists of:

- AOS Custom tools,
- Example software applications,
- Documentation:
  - Hardware & Software Introduction,
  - Quick Start Guide,
  - Detailed Open Source API description, and,
  - Elaborate explanation of software examples.

A snapshot of the *Start Menu* (All Programs) > *AOS* options after installation looks like this:

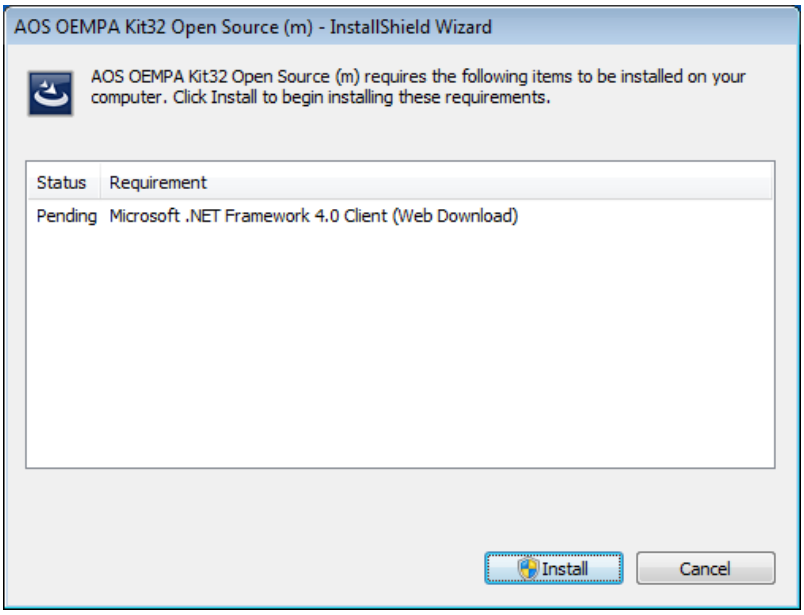


2. Installation

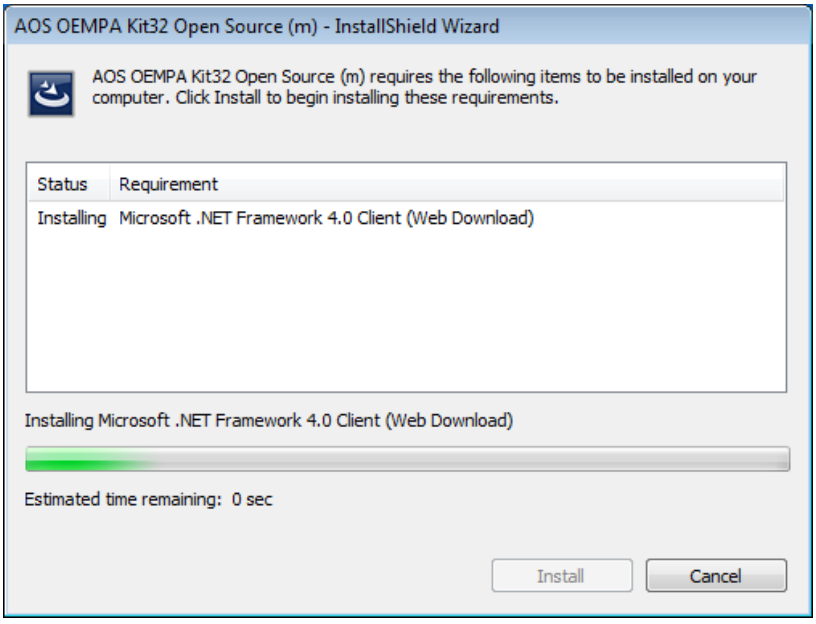
This section outlines the installation of AOS software. The example is for a 32-bit installation, but the procedure is same for 64-bit software version.

2.1 Prerequisites

- Run the executable (.exe) on your system. If errors pop-up, antivirus software might have to be disabled during installation, depending on the antivirus settings.
- The tools require .NET framework 4.0.
    - If not already installed, the installer with install this piece of software first. The window is shown below:



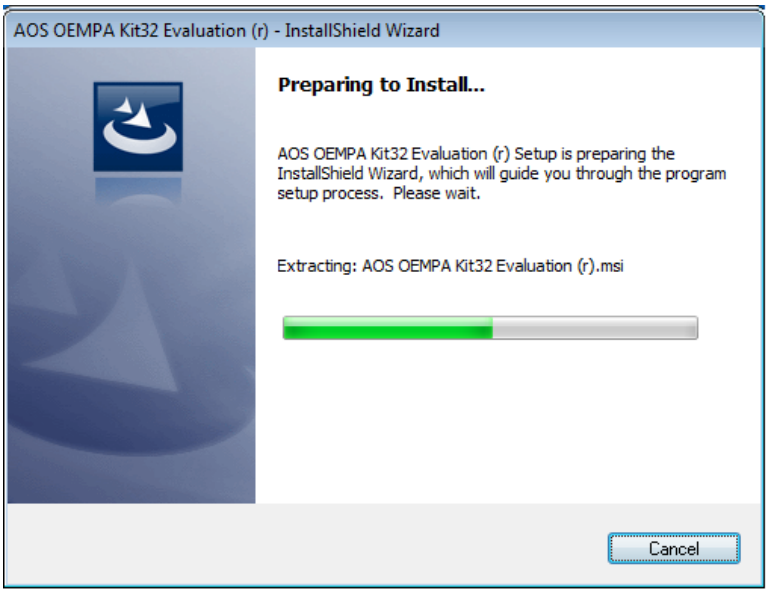
- Click *Install*.



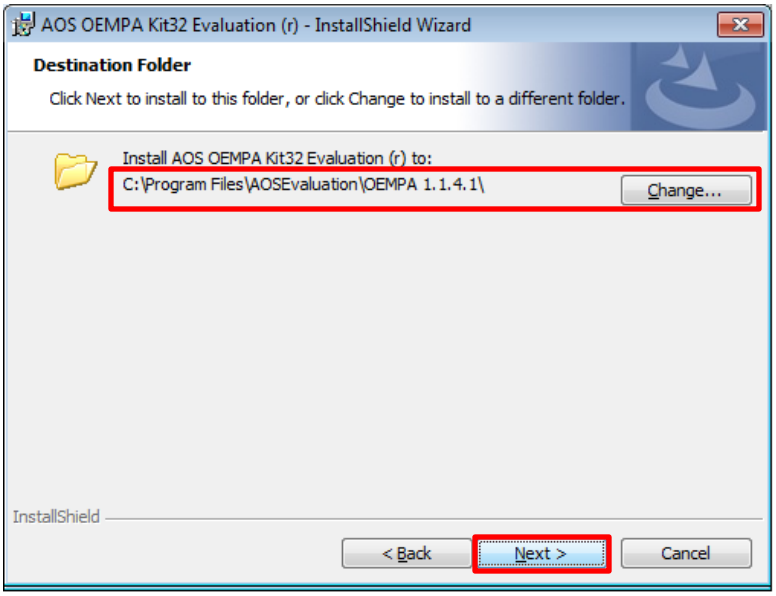
2.2 Software Installation

The installation procedure uses 32-bit Evaluation Kit version 1.1.4.1r as an illustrative example.

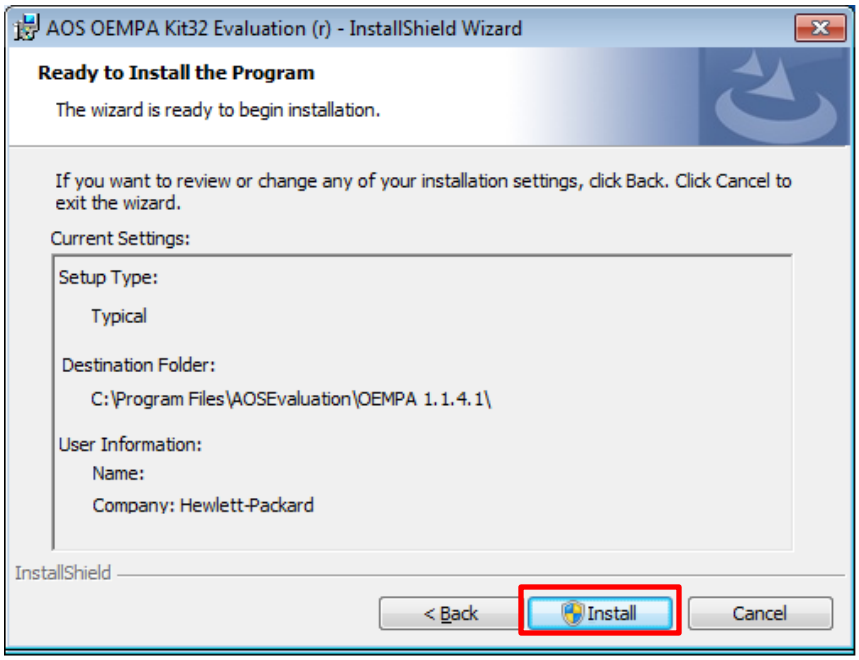
- Run the AOS executable software.



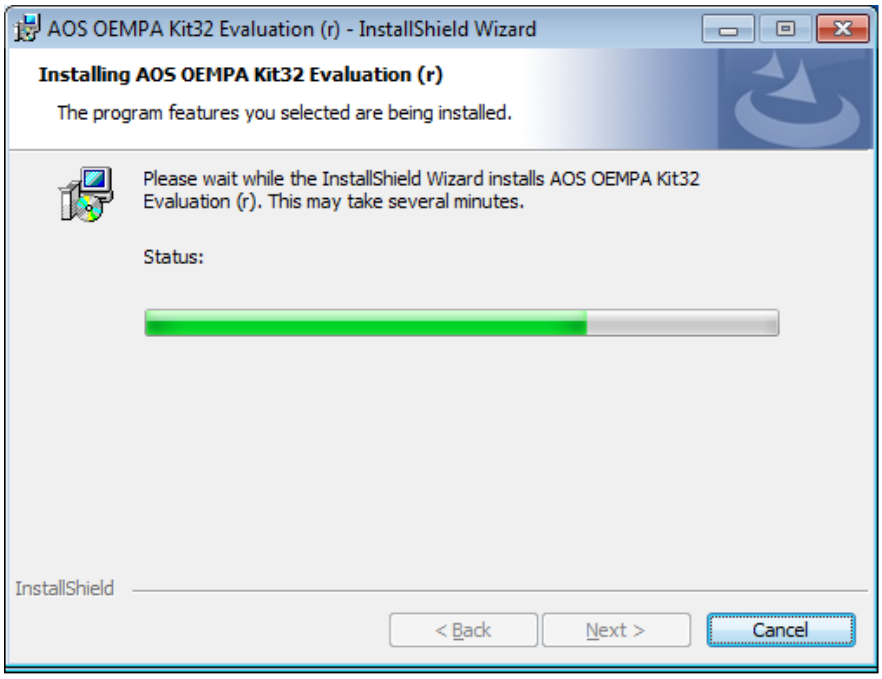
- Choose the desired installation directory.



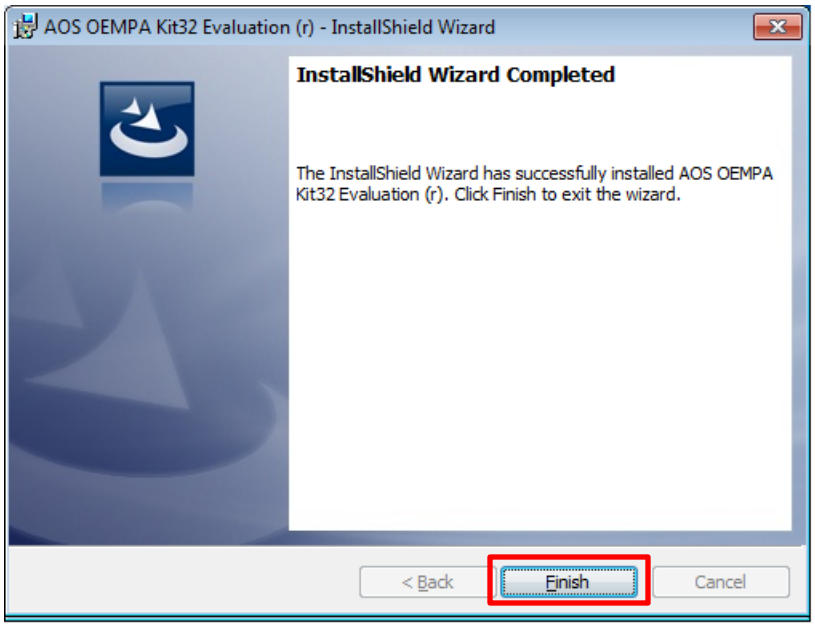
- Click on *Next*.



- Click on *Install*.



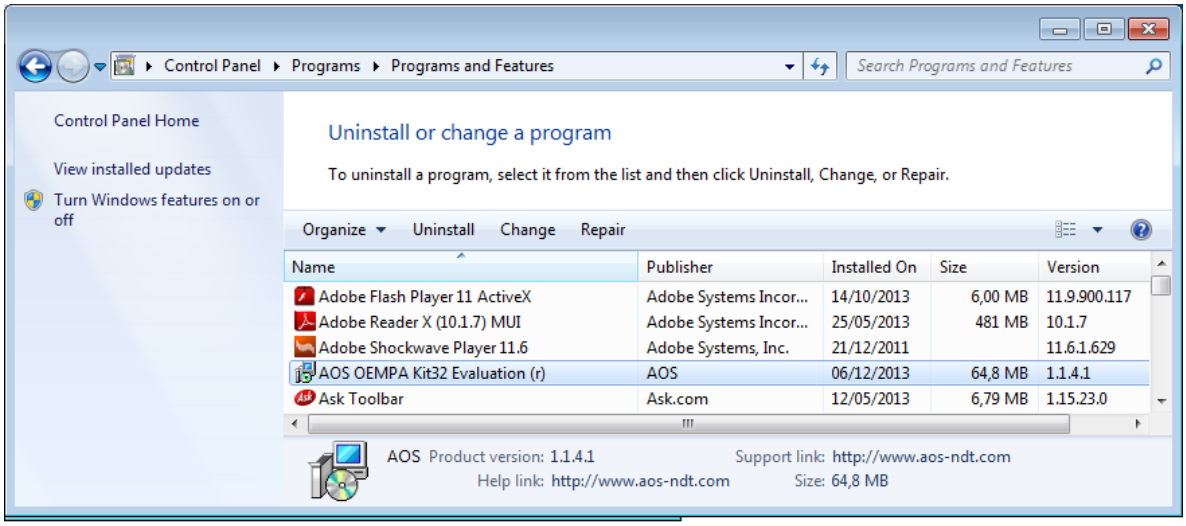
- Click *Finish.*



After clicking on *Finish*, installation requires an additional 30 seconds to be completed.

2.3 Software Uninstallation

AOS SW packages are available in *Programs and Features* for uninstallation as shown below:



Select the AOS software and click *Uninstall*.

Note: The uninstall process removes only files that have been installed and not new files created after installation. For example: OEMPA files, compiler created temporary files in the folder “Sources”. It would be a good idea to delete the folder ‘Sources’ manually.

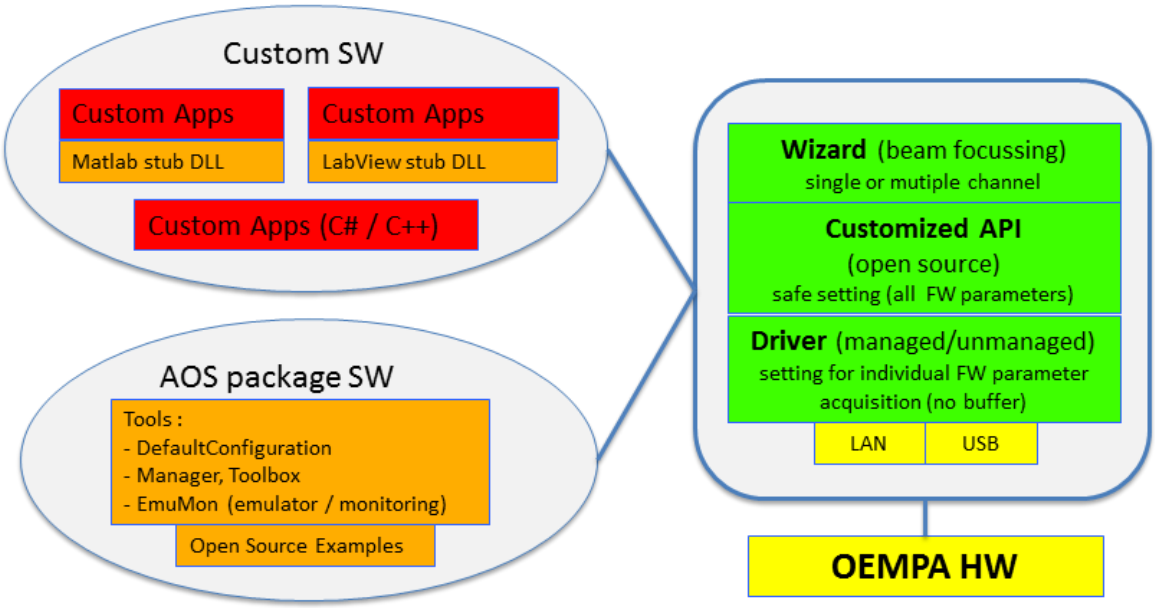
3. Software Architecture

The basic dynamic-link libraries (DLLs) of the package are compatible with any Windows application (MATLAB, LabView etc.). Custom applications can be developed with any programmable language: C++, MFC, C#, etc.

The AOS software is compatible with Windows XP or Windows 7 (32-bit or 64-bit). Development has been completed with Visual Studio 2010, and caution should be exercised when using other versions of Visual Studio.

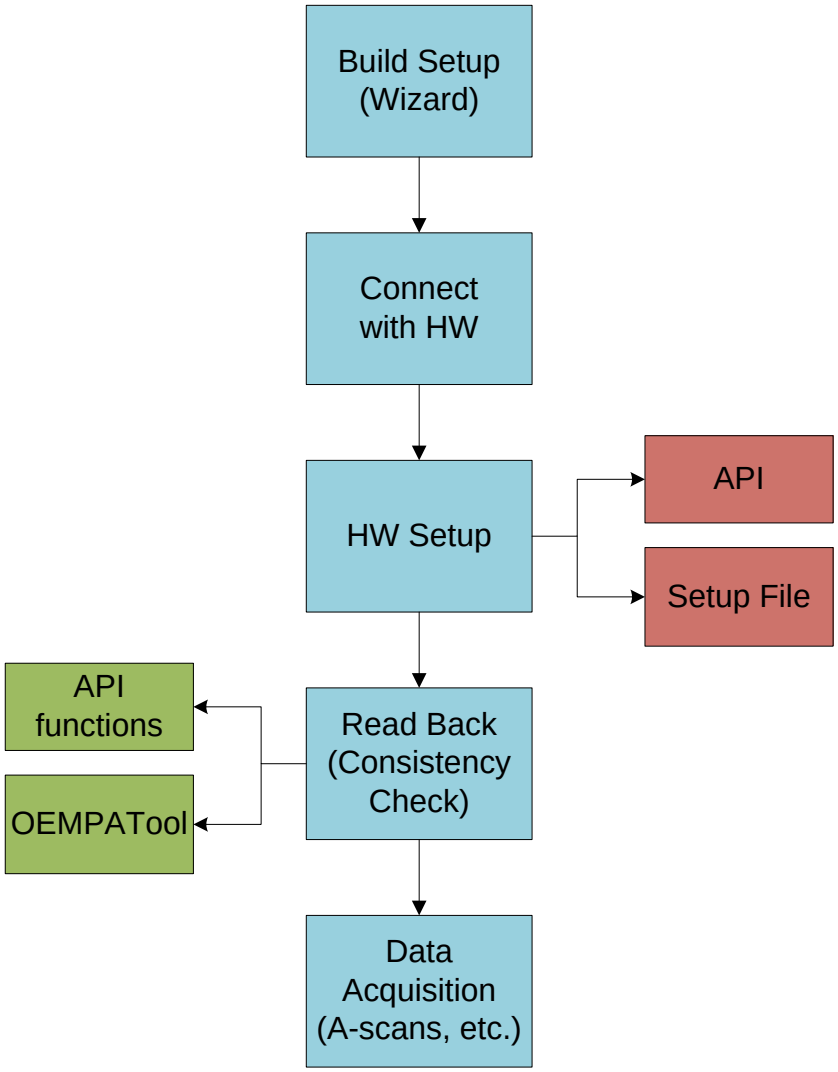
The diagram below outlines the basic architecture of the software and application programming interfaces (APIs).

- The 'Custom SW' bubble on the top left illustrates the development of custom applications by interfacing with the OEMPA API/Driver.
- The 'AOS SW Package' bubble on the bottom left gives examples of helper tools and applications provided to help software developers.
- The LabView driver is not documented here (contact AOS for more details).



3.1 Interfacing with Hardware

Developing custom software to interface with hardware from AOS can be broken down into 5 main processes:



1. **Build Setup** – Create a configuration file to pack all UT and Phased Array parameters (OEMPA file) from the Wizard.
2. **Connection Management** – Manage the communication between software and hardware with a few simple functions.
3. **Hardware Setup** – Send the necessary UT and Phased Array parameters to the hardware from the software.
    a. **Method 1**: Using the OEMPA file. (Often the easiest approach)
    b. **Method 2**: Using the API, where this approach also has a few variants. See below for more details.

4. **Read-back** – Read back the UT and Phased Array parameters that were sent by the software for performing a consistency check of your own software.
    a. **Method 1:** call the required function from your application SW.
    b. **Method 2:** close your application SW, run OEMPATool to perform a read back.
5. **Data Acquisition** – receiving the A-scan data, input events, encoder information, time stamps, C-scan gate results, etc. from the hardware.

3.2 Open Source Applications

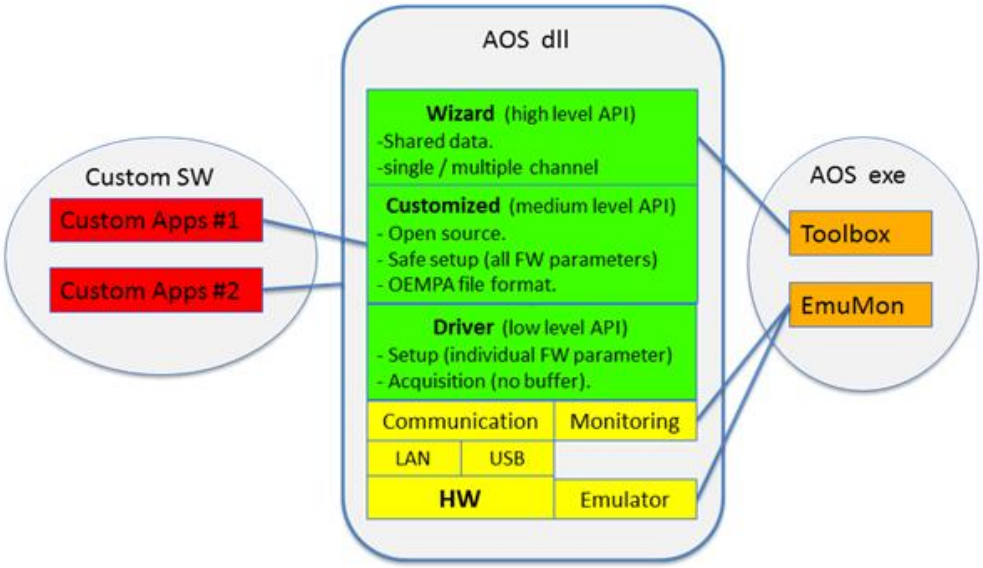
The following software applications are bundled with AOS installations (unless noted, applications are for C++):

1. **Example using Low-level API in C++** ("OEMPAApplicationExample.exe"): Basic example of a custom application SW, demonstrating use of the driver.
2. **Example using Low-level API in C#** ("OEMPAFormExample.exe"): Basic example of a custom application SW, demonstrating use of the driver.
3. **Example using Medium-level API** ("OEMPACustomizedExample.exe"): basic example of using the customized medium-level API. Also useful to understand the internal process of the medium-level API driver.
4. **Example using High-level API** ("OEMPAWizardExample.exe"): basic example of interfacing with the wizard.
5. **OEMPATool** ("OEMPATool.exe"): Example of a custom application SW.
6. **OEMPASector** ("OEMPASector.exe"): Example of a custom application SW.
7. AOS Helper Tools: UT Toolbox, DefaultConfiguration and Manager.
8. **EmuMon** ("UTEmuMon.exe"): Used for debugging, hardware monitoring, hardware emulating, and maintenance.

Before giving more information about each application SW delivered in the different kits, it is necessary to explain basic concepts of the different APIs.

4. Three API levels

The software (SW) has been designed with three different connection points (APIs). In this section, the API is introduced and general information about the software is given.



The low-level API is made of simple functions. The medium-level API is open source, and is made of a basic data structure to pack together all of the FW parameters required to setup the device. This data is referred to as **configuration data**. The high-level API (Wizard) is made of C++ complex classes with links between them.

4.1 Low Level API (Driver API)

- List of simple read/write functions to access individual firmware (FW) parameters.
- Each FW parameter is independent of others. Each function can be used as needed (For example: only update the gain or to upload the full setup).
- Acquisition: Getting the acquisition data stream coming from the FW.
    - No acquisition buffers.
    - Support for multiple acquisition channels.
- As this is a collection of independent parameters, there is no global consistency check between all of the FW parameters as there would be with the high-level API.
- The low-level API can be used to access any low-level FW parameter. Using the low-level API carries some risk, as a single missing FW parameter is enough to freeze the device. In order to properly initialize parameters upon startup, all required FW parameters are conveniently packed together by the medium-level

API to help the developer. After initialization, the low-level API can be accessed to update any FW parameter.

- Data structure to pack all FW parameters is named **configuration data**.

4.2 Medium Level API (Customized API)

API to manage all **configuration data** required by the HW to work properly:

- These DLLs are open source.
  - The customized API can be modified by the user, but would then need to be managed and transferred to new SW updates and releases.
- The customized API driver manages the configuration data.
  - Exchange between the computer memory (RAM), the device, and the hard disk (**configuration files** also named **OEMPA files**).
- The **customized wizard API** is dedicated to the beam focusing only.
  - To exchange **configuration data** with the toolbox.

4.3 High Level API (Wizard API)

The Wizard API is useful for computing focal law delays for many types of samples and probes.

- Single or Multiple acquisition channels for each
  - Focal law delays
    - Plane and cylinder interfaces.
    - Most probes (linear1D, linear2D, matrix for version newer than 1.1.5.0i).
  - Probe specifications (correction)
- Any array has a dynamic size so that future FW improvements will not affect the interface with the Wizard API.
- Data is shared, so the Toolbox can be used in parallel with custom software to edit the inputs.
  - The toolbox display is also useful for the developer to demonstrate the layout of classes and their members.
- Advanced Shared Setup.

4.4 Developer language

The driver and the Wizard have been developed with C++ (unmanaged). There are special stubs for:

- C# (managed): all features.
- MATLAB: main features to use the device.
- LabView: main features to use the device.

For both MATLAB and LabView, source code is provided (Open source) as a foundation for the developer, so it is easier to add new features.

5. Applications

The following applications are included as examples for how to use the low, medium, and high level APIs:

- OEMPAAplicationExample (5.1) – Example using Low Level API in C++
- OEMPAFormExample (5.1) – Example using Low Level API in C#
- OEMPACustomizedExample (5.2) – Example using Medium Level API
- OEMPAWizardExample (5.3) – Example using High Level API

The following is a detailed description of each provided application.

5.1 Examples using Low Level API in C++ /C#

- These are examples of a very simple applications one would develop to interface with an AOS instrument. Source code is provided (Open source) as a foundation for the developer. The documentation includes:
    - Complete examples on how to setup the HW (except creation of the configuration file).
    - Examples of using the low-level API functions
        - Connection with the hardware.
        - Enable/disable pulser.
        - Read/write the “Gain” of the first cycle.
        - Read the “Cycle Count”.
        - Write the synchronization type (internal, encoder or mixed).
    - Examples of using the integrated medium-level API functions:
        - Ability to read/change data before sending it to the hardware.
            - Read the “Gain”.
            - Read the “CycleCount”.
- Displays: throughput of A-scan and C-scan, data lost, and encoders.

5.2 Example using Medium Level API

- This is an example of a simple application one would develop to interface with the medium-level API. Source code is provided (Open source) as a foundation for the developer. Documentation includes:
    - Examples for using the medium-level API functions.
    - Assumption: the OEMPA file has been previously created by “OEMPATool”, “OEMPAWizardExample”, or others...
    - A wrapper of the low-level functions (Medium-level API) is available for copy/paste in application SW. The file name is “CustomizedAPI.cpp” and includes:
        - One global basic data structure to store all FW parameters (configuration data) of an OEMPA device, and some functions to manage it. This structure and its functions essentially make up the medium-level API.
        - After software updates, you will need to update the copy-pasted code.
    - Complete examples of changing **configuration data** that has been loaded from a **configuration file,** before sending it to the hardware. Display/change:
        - Aperture element list.
        - Apodization (useful for normalization) / Pulse width.
        - Beam correction (useful for calibration).
    - The “customized API driver” can be used instead of this medium-level wrapper (see the driver documentation in *Software_API.pdf* for more info).

5.3 Example using High Level API

- This is an example of a simple application one would develop to interface with the Wizard. Source code is provided (Open source) as a foundation for the developer. Having the source code also serves as documentation.
- To create any configuration file that can be loaded by the customized API to properly configure the device.
- Integrated templates:
    - For single acquisition channel: linear scanning only.
    - For multiple acquisition channel scanning/sweeping that is possible with the UT Toolbox.
        - Single probe with multiple scans.
        - Multiple probes with single or multiple scans.

  - Matrix probe support (for version newer than 1.1.5.0i).
- Dialog that helps understand the links between all main items: device, template, Wizard, configuration files, and Kernel files.

6. Utilities

6.1 OEMPATool

- This is an example of an application one would develop to interface to an AOS instrument. Source code is provided (Open source). Documentation includes:
  - Complete examples detailing HW setup.
  - Examples for using the low-level API functions.
  - Examples for using the medium-level API functions (customized API driver and wizard customized API).
- Integrated templates for any simple scanning/sweeping (linear, sector) for single acquisition channel (linear phased array and plane parts).
- Displays (single and multiple acquisition channels):
  - Non-corrected B-scan display (all cycles should have the same point count).
  - A-scan display (minimum and maximum buffers).
  - C-scan display (HW or SW).
  - Throughput, encoders and data lost.
- Access to low-level FW parameters (via the OEMPA text file).
- Debug facility: connect to the device and read back its configuration to confirm that software applications are working as expected.
- Ability to dump in an external text file:
  - The temperatures of the device.
  - The last A-scan of all cycles.
  - The last C-scan of all cycles (amplitude and time of flight).
- Example to show how to remove the Kernel Message Box.

6.2 OEMPASector

Another example of a customized software application.

- Source code provided (Open source)
  - Example of proper HW setup.
  - Examples for using the low-level class API.
  - The medium-level API is accessed through the driver.

- Integrated template for sector scanning/sweeping for simple systems (linear phased array and plane parts) and for single channel.
- Corrected B-scan display, Real-Time C-scan, S-scan, D-scan and A-scan
- Basic offline Analysis Mode
- Access to low-level FW parameters (via the OEMPA text file).
- Matrix probe support (for version newer than 1.1.5.0i).

7. Advanced Utilities

7.1 Hardware and Filter Configuration (DefaultConfiguration)

This software is compatible with any application SW. It must be installed in the same folder as your application SW. This software is used to set:

- Default filters (low-pass, high-pass, band-pass, band-reject).
- Synchronization mode (internal/encoder).

Save them as the default HW configuration, loaded at the first connection time with the device. It will be reused with any of your setups. See the "Hardware and Filter Configuration" chapter in in document *Software_Utilities.pdf* for more details and the "Configuration file" paragraph (4.5.1) in document *Software_API.pdf*.

7.2 Toolbox

- This software is compatible with any application SW. It must be installed in the same folder as your application SW. The Toolbox is the graphical user interface to the "Kernel". An important aspect is the tree-like structure, which is how the object is organized in the source code. It also includes a built-in documentation for help with each parameter.

With the Toolbox you can access:

- Generic Wizard
    - Any probe (mono, 1D linear array, 2D matrix arrays and custom phased arrays for version newer than 1.1.5.0i)
    - Plane or cylinder parts
- Software preferences
    - Automatic connection, disconnection, etc…
    - Language selection (language files are customizable)
- Kernel display.

- Generic high-level UT parameter (device independent)
- Editing of high-level UT parameters (for example, probe element normalization)
- Shared setup: editing of any kernel object of application SW.

7.3 Manager

This software is compatible with any application SW. It must be installed in the same folder as your application SW. Basic functions of the Manager:

- Edit some of the keys in the **software configuration file**.
  - KeepAlive: useful for debugging software.
  - Connection shortcut: to update FW with “EmuMon”.
- Edit options for configuration files (OEMPA file).
- Terminate all other applications (in the current folder).
- Delete all default HW configuration files.
- Delete all default SW layouts.

7.3.1 Layout files

The layout of an AOS application example is saved in a file with the prefix “Layout_” followed by the name of the application. This is a text file saved in folder “C:\ProgramData\AOS\OEMPA [version]\Cfg”. For example, the file is named “Layout_OEMPATool.txt” for the “OEMPATool” SW.

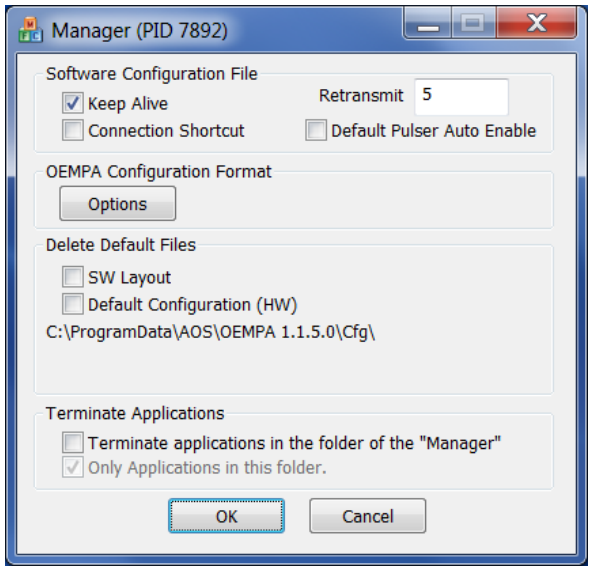
This file is automatically deleted if you change the screen. If there is a problem, you can exit and delete it manually, and then restart the application.

7.3.2 Hardware configuration files

Default HW configuration can be saved with the prefix “HW_”, followed by the name of the application or by the IP address. This is a text file saved in folder “C:\ProgramData\AOS\OEMPA [version]\Cfg”.

You can delete these files with the Manager SW.

7.3.3 Dialog



- "Keep Alive": see paragraph "Debugging".
- "Connection Shortcut": if the device is stuck (Flash erased), this option can help to rebuild the Flash. Please contact AOS.
- "Options": for OEM-PA configuration files.
- "SW Layout": to delete layouts of AOS tools ("OEMPASector", "OEMPATool").
- "Default Configuration (HW)": to delete all default configuration files.
- "Terminate applications in the folder of the "Manager"": terminate frozen or unresponsive software.

7.4 *Emu*lator *Mon*itor (*EmuMon*)

This is a helper tool used for debugging, monitoring the data flowing between the PC and hardware, viewing the hardware memory map, emulating the hardware, and updating the firmware of your device. The connection between *EmuMon* and the application SW is a socket, so *EmuMon* can be run on a different computer (or in any folder).

- HW monitoring and maintenance
    - Display data streams exchanged between the SW and the HW
        - Acquisition and setup data
        - Data streams can be saved and later "replayed" by the emulator, without analysis or knowledge of the stream format.
    - Connection with the driver inside your application SW via one socket
        - Available for any custom application
    - Maintenance of the HW
        - Update the firmware to the latest version.
        - Edit parameters stored in flash memory:
            - IP address, net mask, default gateway, MAC address, board name (used by the driver as a name for your device kernel object), etc.
                - Use caution when changing these values. Preexisting and final values should be recorded to prevent lockout. For example, if the IP address is changed and

forgotten, the device will be unreachable; requiring you to ship the unit to AOS for hard-reset.

- HW emulator
    - To debug custom software applications
    - HW stream breakpoint
        - Breakpoint on each stream (on each sub stream is also possible)

If the software is running on a slow or overburdened computer, and the data throughput is high, EmuMon cannot be used without disturbing the acquisition. In this scenario, it would be best to use the “network protocol sniffer” that has been developed.

8. Tips

The following section contains additional information to use an AOS system:

8.1 Bug Reporting

During execution of any of the AOS programs or while using AOS source code, if any bug should be found, please share the associated file and error report to AOS support for clarification and further updates to the available software.

8.2 Debugging

The following information may be helpful when using the debugger:

- For Ethernet communication, the HW sends KeepAlive messages to the computer to be sure that the connection is still established. The link will be broken if the application developer stays in a breakpoint for more than 10 seconds.
    - A special function is delivered to disable the KeepAlive function (“SetKeepAlive”).
    - The KeepAlive function can also be disabled from the “Manager”.
    - If KeepAlive is turned off, and the connection is broken, either a hardware reboot or software disconnect/reconnect will be necessary to restore the link.
- *EmuMon* (acquisition replay) is useful for debugging acquisition processes.
- A read-back of HW setup parameters can serve as a consistency check to determine whether custom software applications are performing as expected.

KeepAlive process is managed by a specific thread which priority is HIGHEST. If a running process with the same priority takes too much time, then it can disturb the KeepAlive

thread and a disconnection with the device can occur. To avoid this disconnection you can disable the KeepAlive for a while.

8.3 General advice

Disable the pulser before closing the software. This is performed automatically (unless this option is disabled), but may not function properly if the application SW crashes. If this happens, the hardware must be reset.

When all else fails…

- Reset the hardware.
- Reset the software (use “Manager SW”).
- Delete the layout file.

8.3.1 Manager SW

The “Manager” SW can be used to force-quit unresponsive applications. This application will kill all other applications placed in the same folder (don’t forget to check “Terminate applications in the folder of the "Manager"”). If one application freezes, all applications (except Emulator_Monitor) must be terminated.

8.3.2 Hardware error: maximum throughput

When the throughput is too high some data loss can occur, see paragraph “Data loss” in document *Software_API.pdf*.

8.3.3 Error dump

Errors are dumped into a text file. Look in the file below for error messages.

“OEMPA [version]\DumpTraceGlobal.txt”.

8.4 Version Compatibility

Known compatibility issues between different versions.

8.4.1 Driver 1.1.1.0 and 1.0.5.3

The version 1.1.1.0 (and newer) is not fully compatible with the previous version. The function “low-level API” has been removed. A special header is delivered to help migrate from version 1.0.5.3 and older to 1.1.1.0 (see “CompatibilityFunctionAPI_1_0_5_X.h”). In this header you will find preprocessor directives to replace most of the old function low-level API.

8.4.2 EmuMon 1.1.1.0 and 1.0.5.3

Communication protocol between EmuMon and application software has been improved in version 1.1.1.0. EmuMon 1.1.1.0 (and newer) is not compatible with the previous versions of application software. Do not try to use “OEMPATool” 1.0.5.3 (or older) to update the FW with EmuMon 1.1.1.0 (and newer).