

# All-In-One Software Architecture for Wafer Probing Automated Test System

by

Y. Djachiachvili  
Analyst  
Team Specialty Products Corp.  
USA

## Category:

Semiconductor

## Products Used:

LabView™ 6.02  
PCI GPIB IEEE 488.2

## The Challenge:

To design reliable, highly flexible software to perform automated multiple test functions for wafer qualification and for sort tests on different wafer probe stations.

## The Solution:

Using of National Instruments LabView, an object-oriented approach as well as highly modular and messages-based software architecture enables one to implement and handle the complexity of All-In-One (AIO) wafer test.

## Abstract

This paper describes the AIO LabView-implemented software architecture that is designed to control both EG-2001X (Electroglass, Inc.) and Summit 12K (Cascade Microtech, Inc.) wafer probe stations and to perform multiple test functions on a device under the test (DUT). The AIO software allows a user to configure complex probe-motion paths on a wafer, to perform a masked multi-layered mapping of test-function acquisition, and to display real-time results on user selected maps. The high accuracy of Z-height probe positioning is achieved by using an electrical wafer profiling method along with z-height map corrections during the test.

The AIO is designed to meet a high level of versatility, reliability, and ease of adding new test functions. To achieve these goals, an object-oriented approach to designing the AIO software architecture is chosen. The program is composed of six functional modules: user interface module, commands module, motion module, test function module, auxiliary function module, and data storage and display module, all of which are running on separate threads. The commands and data flow between the modules and objects are provided through the messaging network using the “queue-” and “semaphore-” based set of VIs to support the communication protocol.

## Introduction

The process of manufacturing of on-wafer laser diodes involves a costly procedure to test each die (out of thousands on a single wafer) according to customer specifications. The test could include up to ten standard test functions for qualification followed by sorting of the dies according to the requirements. Sequentially performed standard test functions can take tens of hours for a single wafer. Another concern is a limited number of die's contact pad touchdowns by a contact probe(s) that supplies an excitation current to the

DUT. Therefore it is highly desirable to perform as many test functions at a single pad's touching as possible to decrease the cost of the qualification phase in the laser dies manufacturing.

Emcore Corp. (Albuquerque, NM) has proposed (credit Dan Jensen) a new approach for on-wafer laser dies characterization test by performing multiple standard test functions on each die of the wafer during a single probing. This approach also provides a higher quality of die the test since each test is performed at the same condition.

Team Specialty Products Corp. (TSP) was contracted to develop software that implements AIO wafer tests on existing hardware architecture, providing full automatic control of both EG2001X (Electroglass, Inc) and Summit 12K Series (Cascade Microtech, Inc) wafer probe stations.

## Hardware Architecture

The hardware architecture is shown in Figure 1. It incorporates wafer probe station 1 or 2 and instrumentation hardware 3 to perform different test functions. As can be seen from the Figure 1, the AIO measurement system can be used with EG2001X or Summit 12 K wafer probe stations. These probe stations have different communication interfaces GPIB and DDE for EG2001X and Summit 12 Cascade respectively. The EG2001X system configuration consists of two NI PCI-GPIB controllers. The first controller (GPIB0) is dedicated to communicate with the probe station while the second one (GPIB1) provides data acquisition and chuck temperature control.

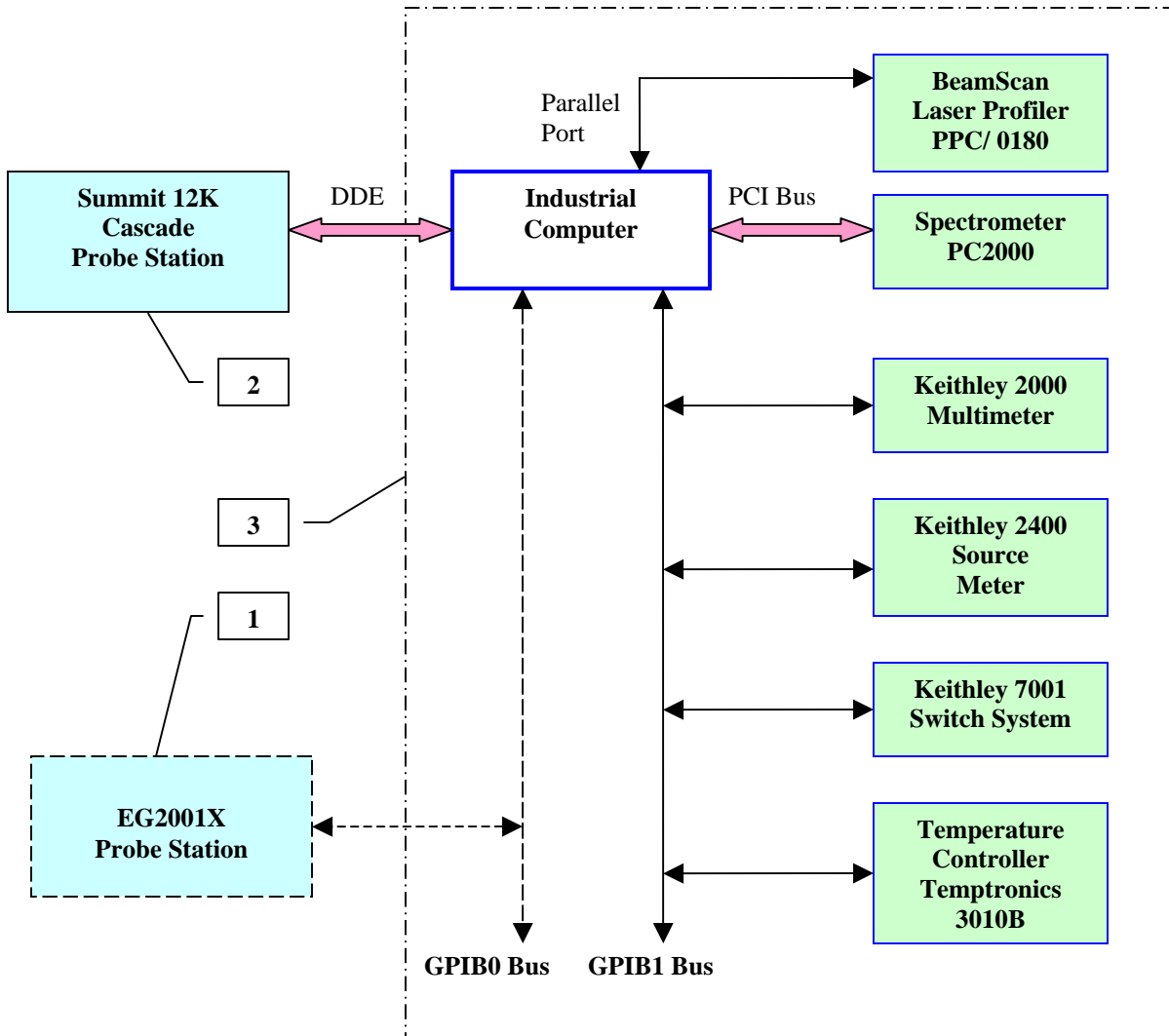


Figure 1. Hardware Architecture of the AIO Multiple Test Functions Wafer Probing System

## Software Architecture

In order to correctly address customer's requirements, the *conceptual view* of the software architecture was developed. In the *conceptual view*, problems and solutions are primary in the application domain terms that originated from the software requirements. The software was decomposed into the high-level components of the system and the relationships among them were identified. The software consists of six major components the User Interface, the Commands Engine, the Data Display and Storage, the Auxiliary Functions, the XYZ Motion, and the Test Functions. These components are shown in Figure 2 as square green boxes and are implemented as independent modules that are executed on separate threads (multithreading is beautifully supported by LabView).

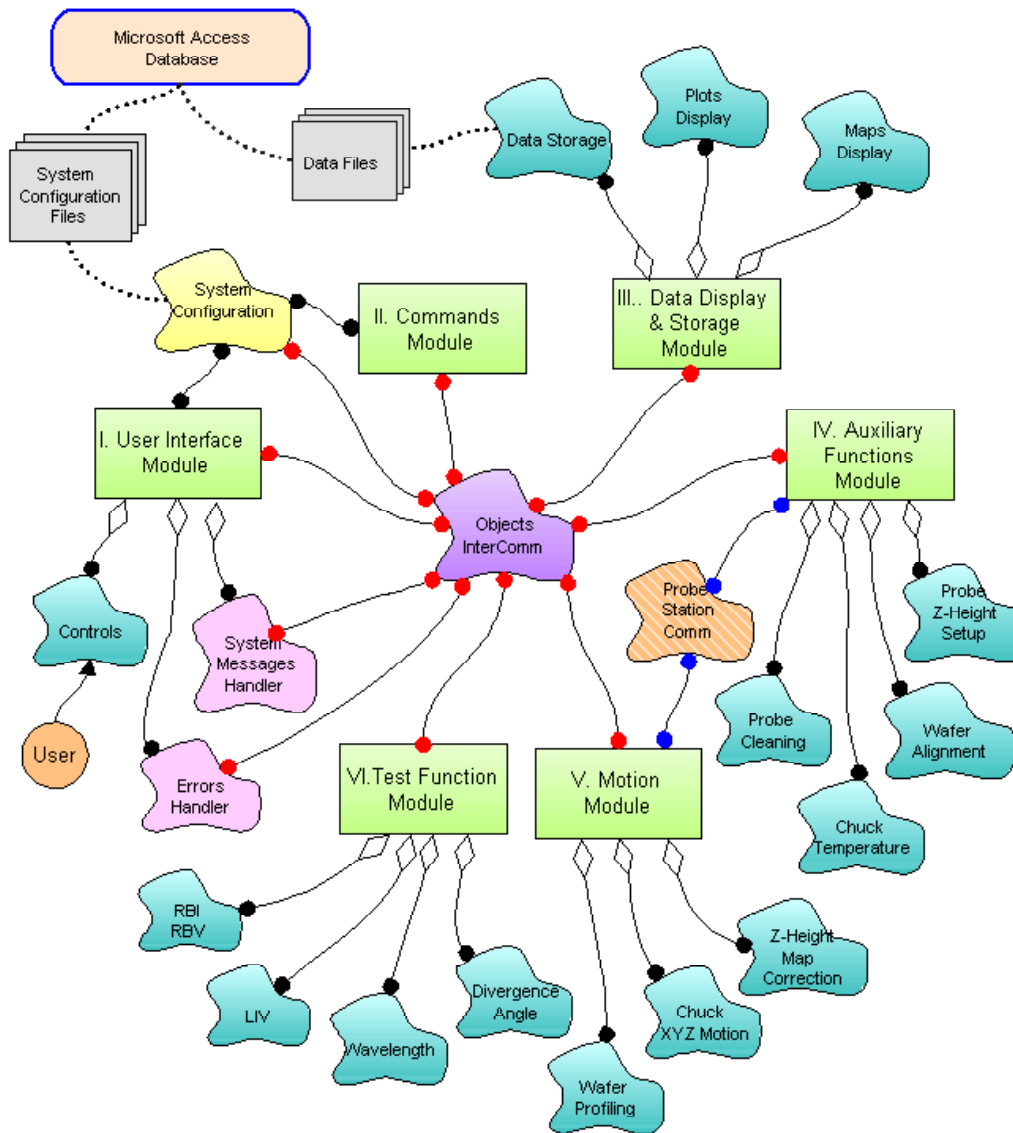


Figure 2. Software Architecture. Objects and Modules Relationships

This decomposition was done in order to address the complexity of the project, to maximally decrease the coupling between the components, and to provide flexibility for future modifications. Thus, after this first

part of principle “divide and conquer” was implemented, the next problem arose. How to organize the commands and data transport between the modules without destroying the objects’ decoupling? The solution was found by constructing the Inter-Object Communication (IOC) Class, which is the central part in the objects and modules relationship diagram (see Figure 2). The main function of IOC is to provide a virtual link between the modules and objects. It utilizes a message-based network to transport commands and data but also includes a mechanism that provides modules or objects synchronization to perform simultaneous operations (or group of operations) in the different modules. The IOC allows dynamic linking (or re-linking) of modules and objects during the run-time by sending the specially formatted packages of bounded messages to the linked objects and modules and then awaiting a command to trigger the simultaneous execution or to perform a pre-defined chain of operations (this mechanism will be explained in great detail in my presentation on NIWeeks 2003: *Modules Synchronization In Message-Based Architecture of LabView Applications*). During the program execution each object or module is allowed to asynchronously send or receive a command(s) or data. The IOC uses the “queue-“ and semaphore-“ based set of VIs to transport messages and data between the objects and to prevent locking or trapping messages inside the queue’s stack respectively.

Figure 2 represents the *structural view* of AIO software architecture and shows the objects’ relationships and their associations with the modules. As can be seen, the Probe Station Communication (PSC) Class is linked to the Auxiliary Functions Module and Motion Module only. The interface between PSC object and Modules allows using a completely different set of commands to control EG2001X or Summit 12K probe stations. The actual link to the particular probe station interface and commands set is performed during the system configuration.

The exceptions, errors, and system messages are handled by the Errors and System Messages Objects that are aggregated with the User Interface Module but have an independent support from the IOC class. The test function objects have a “has” type association with the Test Function Module and its implementation allows to externally re-program the AIO software to run built-in test functions in any arbitrary test scheme. The system configuration is provided through the Microsoft Access Database via the system configuration files. All acquired data are stored locally as ASCII files and can be accessed through the network.

## Conclusion

We showed that the LabView could easily handle the design of large-scale applications that includes multi-threading and object-oriented approach. It also reliably supports the use of message-based software architecture that dramatically increases the design flexibility of LabView applications and produces a very “clean” code diagram with transparent relationships between the objects and modules. Example of AIO G-2 LabView diagram is shown in Appendix.