UCRL-JC-124381
PREPRINT

A Prototype Distributed Object-oriented Architecture for Image-based Automatic Laser Alignment

Eric A. Stout
Victoria J. Miller Kamm
James M. Spann
Paul J. Van Arsdall

This paper was prepared for submittal to the
2nd Annual Solid-state Lasers for Applications to Inertial Confinement Fusion Conference
Paris, FRANCE
October 22-25, 1996

October 15, 1996

A prototype distributed object-oriented architecture for image-based automatic laser alignment

Eric A. Stout
Victoria J. Miller Kamm
James M. Spann
Paul J. Van Arsdall

Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, California 94550

ABSTRACT

Designing a computer control system for the National Ignition Facility (NIF) is a complex undertaking, because of both the system's large size and its distributed nature. The controls team is addressing that complexity by adopting the object-oriented programming paradigm, designing reusable software frameworks, and using the Common Object Request Broker Architecture (CORBA) for distribution.

A prototype system for image-based automatic laser alignment has been developed to evaluate and gain experience with CORBA and OOP in a small distributed system. The prototype is also important in the evaluation of alignment concepts, image processing techniques, speed and accuracy of automatic alignment for the NIF, and control hardware for alignment devices. The prototype system has met its initial objectives, and provides a basis for continued development.

Keywords: object-oriented, CORBA, frameworks, automatic alignment

1. THE NATIONAL IGNITION FACILITY CONTROLS ARCHITECTURE

The computer control system for the National Ignition Facility (NIF) must be designed to meet several significant challenges. First and foremost, it must manage the size and complexity of the machine: the NIF will be a 192 beam laser containing approximately 48,000 control points, controlled by several hundred computers running several hundred thousand lines of software. Furthermore, the control system must be designed in anticipation of the long lifetime of the facility: over the course of thirty years, every computer in the system will be replaced at least once, and specific controls and diagnostics will come and go. The design of the control system should facilitate this evolution.

1.1 The object-oriented paradigm

The use of the object-oriented paradigm is the key to controlling the complexity of the control system. According to Booch, "By applying object-oriented design, we create software that is resilient to change and written with economy of expression."[1] The principle of object-oriented design is that a software system consists of a collection of objects, each with its own set of behaviors, which collaborate to fulfill the functions of the system. Each object may maintain internal state information which may only be affected from without by invoking one of its behaviors (or operations). The type of an object is referred to as its *class*. Classes which have attributes in common may *inherit* those attributes from a common "parent" class.

There are several programming languages which offer these object-oriented features, including C++, Smalltalk, and Ada95. Ada95 has been chosen for the NIF control system primarily for its reliability. Barnes says: "It is now clear after many years' use that Ada is living up to its promise of providing a language which can reduce the cost of both the initial development of software and its later maintenance."[2] For such a large system to be produced in a timely fashion and to be maintainable over its long lifetime, Ada95 - an update of the original Ada specification, with the addition of object-oriented features - is an excellent choice.

1.2 Reusable software frameworks

A framework provides a set of related services which are required throughout a software system. Every application which requires a particular service can use the same framework, specializing it by inheritance if necessary but reusing much of the code.

There are a number of such frameworks in the NIF software design for such applications as message logging, status monitoring, sequence control, and configuration. The Sequence Control framework provides facilities for creating and executing command sequences, or scripts. Describing complex operations which may be performed in multiple ways (such as laser alignment) via scripts allows those operations to be modified without recompiling software. The Configuration framework maintains a hierarchy of the computer-controlled devices in the NIF, along with initialization information for each device and an “address” at which that device may be located for control. Reusing these frameworks, rather than reimplementing them in each application, allows developers to devote their time to implementing the features unique to their particular domains.

1.3 The Common Object Request Broker Architecture (CORBA)

Adopting the object-oriented approach and using frameworks to enable software reuse are strategies which mitigate the general problem of software complexity, but the fact that the NIF control system is distributed among many computers is not addressed by these strategies. Fortunately, tools are available which facilitate distributed computing. The two major standards applicable to the NIF are the Distributed Computing Environment (DCE)[3] and the Common Object Request Broker Architecture (CORBA)[4]. Because CORBA provides direct support for object-oriented programming in a distributed system, it has been selected for use on the NIF. The source of the CORBA standard is the Object Management Group (OMG), a consortium of some 500 companies interested in standardizing interactions between distributed objects.

The fundamental principle of CORBA is that an object on one computer should be able to call an object on another computer in just the same way as if both objects were on the same machine. A lot goes on behind the scenes to route the call to the right destination and translate between the different machines’ data representations, but all the complexities of network programming are hidden from the developer. Simply put, CORBA acts as a software bus between objects.

1.4 A layered control system

Object-oriented programs have been described as “object soup,” because a representation of the program’s runtime behavior often looks like a tangle of objects sending messages back and forth to perform the program’s function. To some extent such a tangle is inevitable, but in a large system some policy must be imposed which dictates the extent of the control authority of each application and its associated objects. This policy is expressed as a set of layers. In the specific case of the NIF control system, there are seven layers: director, supervisor, application front end processor (FEP), service FEP, embedded controller, permissive, and device (see Figure 1), Objects in a given layer may issue control commands to objects in layers which are both below and adjacent to their own. Thus, an application FEP may command a service FEP, but not a supervisor or another application FEP. A service FEP may command only the devices which are physically attached to it, and their associated permissives, where present. In this manner, an orderly flow of control is preserved. A service FEP is an FEP with no autonomous function: it merely executes commands as they are received. By contrast, an application FEP has a more complex function, which, once initiated, the FEP will carry out without further instruction.

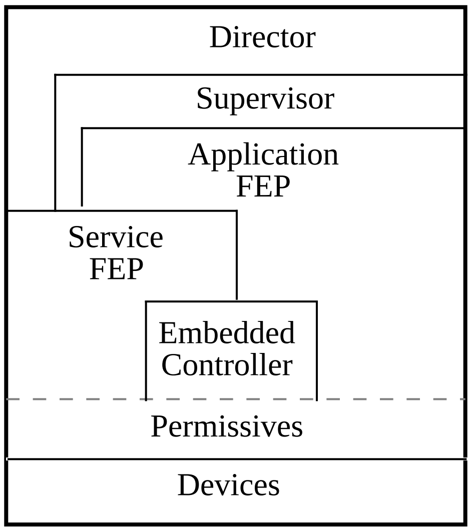

Figure 1. NIF architecture layers

2. PROTOTYPE AUTOMATIC ALIGNMENT SYSTEM HARDWARE

The prototype automatic alignment system is a part of the NIF Alignment Concepts Laboratory, which contains a 1:10 scale model of a NIF beamline (see Figure 2). Controls experiments are conducted in the transport spatial filter (TSF) area of the beamline. A mirror has been inserted between the TSF and the cavity spatial filter (CSF) so that experiments may be conducted simultaneously in both areas.

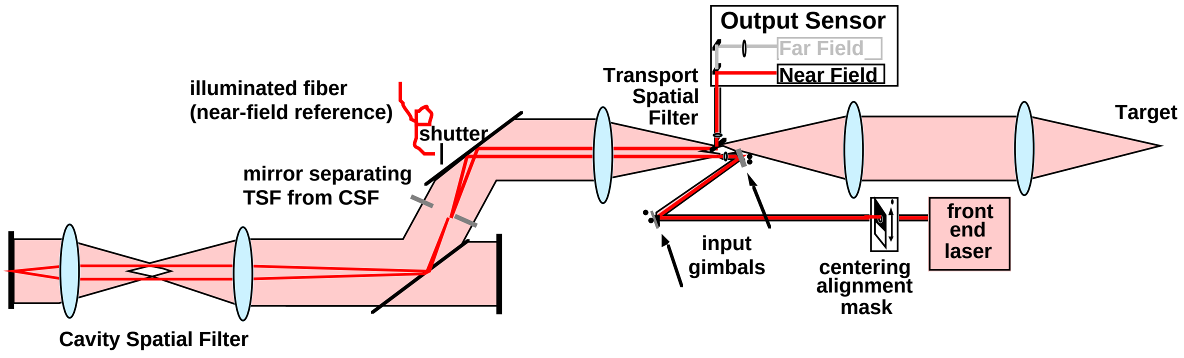

Figure 2. The Alignment Concepts Lab model beamline

The prototype system includes an automatic alignment (AA) FEP that receives input from cameras in the beamline, and an alignment controls (AC) FEP that controls alignment devices. Each FEP consists of a VMEbus chassis controlled by a SPARC CPU. Additional VME cards provide specific device control and processing functions. Both FEPs are networked via Ethernet, so that they may communicate with one another and also with software running on other networked computers. A block diagram of the system is shown in Figure 3.

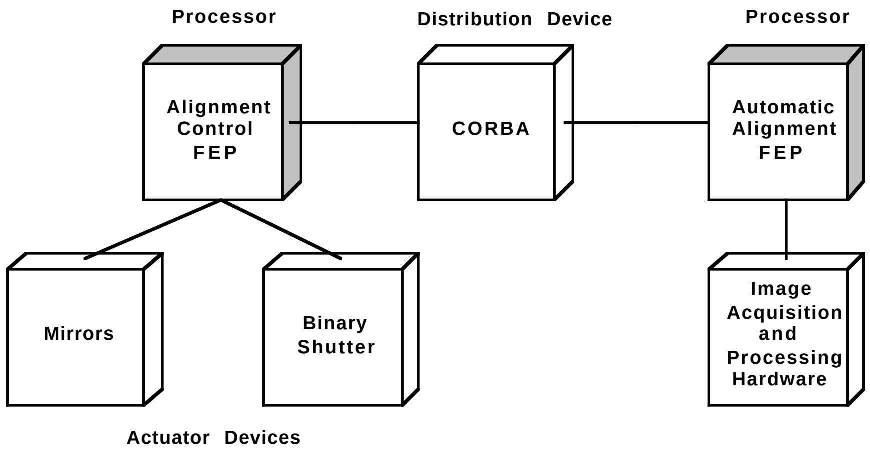

Figure 3. Booch process diagram showing two Front End Processors and attached devices collaborating via CORBA.

2.2 Prototype AC FEP hardware

The AC FEP is responsible for control of alignment devices, including motorized mirrors, translation stages, and shutters. To this end, the prototype AC FEP contains two cards in addition to the CPU: a six-axis motor indexer produced by Oregon Micro Systems[5], and a digital I/O card with 16 binary outputs produced by Themis Computer[6]. Currently the AC FEP controls the input gimbals - two motorized mirrors which steer the injected laser beam - and a shutter which either blocks or passes light from a fiber light source which provides a near field reference for centering the beam.

2.3 Prototype AA FEP hardware

The AA FEP determines, on the basis of images obtained from CCD cameras, the alignment error at a given stage of a laser beam chain by comparing the beam location to a previously acquired reference. It then requests a motor movement from the AC FEP to correct that error. The prototype AA FEP includes cards which digitize, store, and analyze images. Near field and far field images of the injected beam are provided by two CCD cameras located in the output sensor area of the model beamline.

2.3.1 MaxVideo 200

The MaxVideo 200[7], produced by Datacube Inc., is a configurable base card for an image processing system. Various modules can be plugged into the card to perform particular image processing functions, depending on the application, and data can be passed to and from separate cards which support Datacube's MaxBus architecture. In the prototype AA FEP, the MaxVideo 200 is configured for simple image acquisition and staging. An analog input module provides four channels of video input. Analog images are converted to 8-bit grayscale and stored in a memory module, from which they may be retrieved by the host CPU or read by another image processing module or card.

2.3.2 APA512+

The actual image analysis in the prototype is performed by Atlantek Microsystems' APA512+[8], an "area parameter analyzer." The APA512+ obtains an image from the MaxVideo 200 and converts it from 8-bit grayscale to 1-bit black and white, based on a user-supplied threshold value: pixels below the threshold are black and those above the threshold are white. It then scans the image for "blobs:" groups of connected pixels of the same color. The user may specify size and color criteria which a blob must meet in order to be reported. The APA512+ reports various statistics pertaining to each blob, including its centroid. Because the image is processed in black and white, the centroid of a blob is equivalent to its geometric center.

3. PROTOTYPE AUTOMATIC ALIGNMENT SYSTEM SOFTWARE

The prototype system represents the bottom layers of the layered controls architecture: the AC FEP is a service FEP, the AA FEP is an application FEP which uses the services provided by the AC FEP, and both FEPs control some devices.

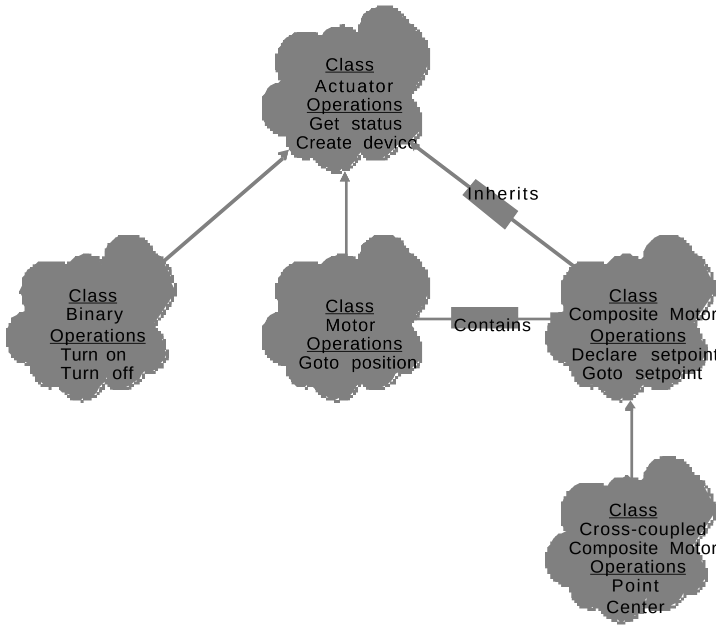

Figure 4. Alignment Controls FEP class diagram

3.1 AC FEP software

The AC FEP is extremely well suited to the object-oriented approach, because its control functions are all attached to physical objects whose relationships are easy to describe. The prototype AC FEP was our first object-oriented Ada95 project. It defines several classes of devices which may be controlled, all of which inherit from the parent class, "actuator." Objects of class "motor" are responsible for control of a single motor. "Composite motor" objects control devices which contain between one and four motors. A "cross-coupled composite motor" is a special kind of composite motor which contains four motors whose movements are not independent; cross-coupling is described in more detail below. In the object-oriented terminology described previously, the class "cross-coupled composite motor" inherits from the class "composite motor." "Binary" objects control devices of various kinds which have only two states, e.g. shutters and light sources. Figure 4 is an object-oriented model of these classes and their relationships.

3.1.1 Cross-coupling

Using a single mounted mirror with two axes of adjustment, a ray may be directed to pass through a single point of interest past the mirror; its line of travel is determined by the point at which the ray strikes the mirror and the point of interest through which it passes. Using two such mirrors, however, the ray may be directed to pass through a selected point in each of two planes of interest, within the constraints of the mirrors' range of motion. If only one of the mirrors is moved, the point at which the ray passes through both planes will change. If both mirrors are moved appropriately, however, one of the two points may be held fixed while the other is moved. We refer to such interdependence as cross-coupling. The cross-coupling relationship of a pair of mirrors each with two axes of adjustment may be described by a 4x4 matrix of real numbers. Multiplying this cross-coupling matrix by a column vector denoting the desired (x, y) adjustment in each of the two planes of interest yields a column vector denoting the required adjustment of each of the four axes. The most common cross-coupled operations in the laser alignment process are "pointing" and "centering." Pointing is the alignment of the laser in the far field, and centering is alignment in the near field.

3.2 AA FEP software

The prototype AA FEP executes pointing and centering loops on the injected laser beam by acquiring and processing images using the hardware previously described, and issuing cross-coupled movement commands via CORBA to the input gimbals controlled by the AC FEP based on those images (Figure 5). A graphical user interface (GUI) is provided so that the progress of a loop may be observed. Additionally, because not all of the devices which must be manipulated to set up each loop are controlled by the prototype AC FEP, the GUI allows the user to initiate the loop after the manual setup is completed.

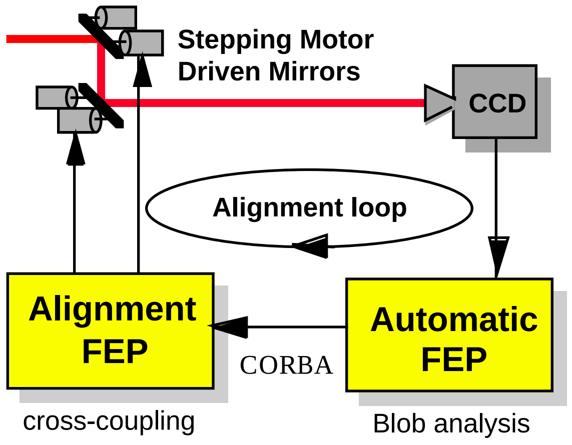

Figure 5. An alignment loop

3.2.1 Alignment loops

Laser alignment is performed as a series of alignment loops. Four kinds of loops have been identified for the alignment of the NIF: pointing, centering, beam focusing, and beam rotation. The four differ in their particulars, but they all have the same basic structure. Pointing and centering loops begin with the acquisition of a reference location, either by acquiring and processing a reference image or by retrieving a saved value. The following steps are then repeated until the location of the beam matches the location of the reference:

AA FEP:

Acquire and process an image of the beam

Calculate the difference between the beam location and the reference location

```
        If the difference is greater than the required tolerance, issue a motor movement correction command
    AC FEP:
        Translate the correction command to motor steps using the cross-coupling matrix
        Move the motors
    AA FEP:
        Wait for motor movement to be completed
When a loop is completed successfully, the next loop in the series can begin.
```

The loops performed by the prototype are a centering loop which centers the injected beam on a mirror at the end of the TSF, and a pointing loop which directs the beam through a pinhole at the focal plane of the TSF.

4. RESULTS

The purpose of the work on the automatic alignment prototype system to this point has been more to provide a proof-of-principle than to provide quantitative results. Prior to the construction of the prototype, however, measurements were taken to determine the performance of CORBA. The prototype itself demonstrates that automatic alignment can be performed efficiently by matching the location of an alignment image to the location of a reference image, subject to the stability of the beam. Moreover, the prototype provides the first demonstration of distributed device control using CORBA and Ada95.

4.1 CORBA performance

We devised several small-scale tests to gain an understanding of how CORBA might perform based on three different conditions: amount of data being transferred with each call, whether the remote object is located in a different process on the same machine or on a different machine, and whether the network and CPU are lightly or heavily loaded. Tests were performed using Orbix 1.3, a CORBA implementation produced by Iona Technologies[9]. The results, summarized in Table 1, suggest that CORBA will be able to satisfy the control data flow requirements of the NIF.

	Lightly Loaded		Heavily Loaded	
Message Size	2 machines	Same machine	2 machines	Same machine
25 bytes	0.5 ms	0.2 ms	7 ms	0.2 ms
100 bytes	3 ms	1 ms	60 ms	2 ms
10,000 bytes	30 ms	10 ms	200 ms	16 ms

Table 1. CORBA performance test results

Note that messages of the latter two sizes were actually passed twice in each transaction: the receiving object merely returned the message to the sender. The amounts of data passed in each transaction, therefore, were 200 bytes and 20,000 bytes. In the "same machine" tests, we observe up to a 100% increase in transaction time between a lightly loaded machine and a heavily loaded machine. Network loading appears to be a much more significant factor in determining transaction time, as transactions between two machines took up to 20 times as long on a heavily loaded network than on a lightly loaded network. A 10 Mb/s Ethernet network, subject to slowing as a result of collisions when heavily loaded, was used for these tests. The NIF will use an Asynchronous Transfer Mode (ATM) network, a much faster switching network on which collisions do not occur.

4.2 Performance of the automatic alignment prototype

Beam pointing and centering have been successfully demonstrated in the Alignment Concepts Laboratory. Successful completion of an alignment loop is defined as matching the location of the alignment beam to the location of the reference spot to within 1.0 pixel. The time required to successfully complete each loop has not been accurately measured: there is an excess delay of between one and two seconds between the completion of one loop iteration and the start of the next, because the software delay function used to wait for motor motion to complete provides a minimum increment of one second. One measurement which has been made, however, is the number of iterations required for successful loop completion. The results for pointing and centering differed considerably. The centering loop almost invariably completes successfully in two loop iterations: that is, on the third iteration through the loop, the beam location is within 1.0 pixel of the reference location. The pointing loop is much less reliable, because the far field (pointing) beam image is not stable in the present laboratory environment: viewed on a monitor, the image of the far field beam visibly "dances" within a range of about two pixels. With such an unstable beam, an alignment accuracy requirement of 1.0 pixel is not reasonable, and meeting the requirement on a given loop iteration is a matter of luck rather than skill. The centering beam image, on the other hand, is quite stable: repeatedly acquiring the centroid of the centering spot shows a short-term drift on the order of 0.3 pixel or less, and its motion is not visible on the monitor.

The NIF facility will be much more carefully controlled with regard to vibration, temperature, and air flow than is the Alignment Concepts Laboratory. We attribute beam movement in the laboratory to these physical phenomena, and conclude as a result that the alignment concept we have demonstrated will meet the NIF requirements under the NIF conditions.

5. CONTINUED PROTOTYPE DEVELOPMENT

The prototype automatic alignment system will continue to serve as a testbed both for automatic alignment in particular and for the NIF computer controls architecture in general. Having successfully executed single pointing and centering loops, the prototype will be extended considerably in order to demonstrate that an entire beamline may be aligned using the same concepts and then to show that many beamlines may be so aligned by a single AA FEP within the NIF time guideline of 30 minutes. As the prototype is extended, and as prototypes of the of the NIF software frameworks are developed, the use and specialization of the frameworks by the automatic alignment system will also be demonstrated.

Extending the prototype to control the entire Alignment Concepts Lab beamline primarily requires that the prototype AC FEP be expanded to control more devices (roughly 20 motors, rather than the current 4) and that an adequate interface be provided for manual operator control of those devices. The prototype AA FEP software must be generalized somewhat to allow for the alignment of the additional loops. The extension to an entire beamline provides an excellent opportunity to incorporate and test prototypes of the Configuration and Sequence Control frameworks. The Configuration framework will be used to create and initialize the AC FEP's device objects at system startup and then to provide the AA FEP with references (addresses) to those objects. The Sequence Control framework will be used to define the set of loops to be aligned, and the order of their execution.

The 192-beam NIF design contains roughly 4,000 alignment loops. The current control system design calls for four AA FEPs to execute those 4,000 loops in 30 minutes. Serial execution on each of the four FEPs will not meet the requirement. However, the execution time of a given loop is dominated by the time required to complete a motor correction command. Multi-tasking software will allow a single computer to perform automatic alignment of multiple beams in parallel to achieve the 30 minute goal. Although we now have only a single beamline to align, software could be written to simulate parallel alignment of multiple beams.

Finally, since this prototype automatic alignment system was constructed, the NIF design for video distribution has changed considerably. In place of an analog switching system that delivers analog images to be digitized by the AA FEPs, the current design calls for dedicated Video FEPs to acquire and digitize images and pass the digital images over the computer network. In addition, cost-performance analysis of computer workstations versus specialized image processing hardware suggests that it is more cost-effective to perform image processing with software on a fast workstation than to use specialized hardware. As a result, the next-generation AA FEP prototype is likely to be a standard multiprocessor server, rather than the current VMEbus system with image digitization and processing hardware.

6. SUMMARY

Prototyping continues to be an important component of the development strategy for the NIF control system. The automatic alignment system prototype has provided valuable initial experience with and validation of the tools that will form the basis of the control system: CORBA and OOP. The prototype has been integrated with a model NIF beamline to demonstrate the basic concepts of NIF automatic alignment. For those reasons, we consider the prototyping effort to have been a success to this point. In keeping with the iterative strategy of software development being used for the NIF, the prototype system will change to reflect changed system specifications and will grow to perform more complex tasks and to more completely represent the NIF software architecture.

7. DISCLAIMER

This work was performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

8. REFERENCES

[1] Grady Booch, *Object Oriented Analysis and Design with Applications*, p. 24, The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994
[2] J. G. P. Barnes, *Programming in Ada Plus an Overview of Ada 9X,* p. 6, Addison-Wesley Publishing Company Inc., 1994
[3] Shirley, J. et al, *Guide to Writing DCE Applications*, 2nd Edition, O'Reilly & Associates, Inc., 1994
[4] *Common Object Request Broker: Architecture and Specification*, Object Management Group and X/Open, OMG.91.12.1, 1991
[5] *User's Manual, Intelligent Motor Controllers, VMEX Family,* Oregon Micro Systems, Inc., Beaverton, Oregon, 1991
[6] *TSVME411User's Manual*, Themis Computer, Gieres, France
[7] *MaxVideo 200 Hardware Reference Manual*, Datacube, Inc., Danvers, Massachusetts, 1993
[8] *APA512+ Hardware Manual*, Atlantek Microsystems Pty. Ltd., Adelaide, SA, 5095, Australia, 1993
[9] *Orbix 1.3 Programmer's Guide,* Iona Technologies Ltd., Dublin, Ireland, 1995